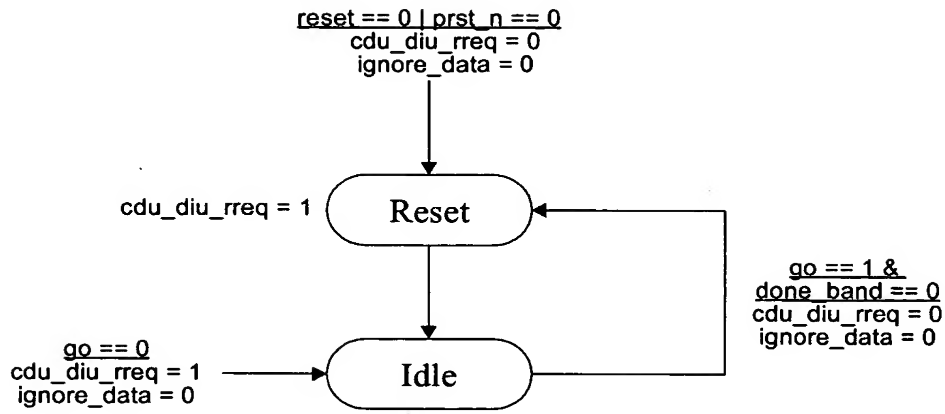

reset == 0 | prst_n == 0
cdu_diu_rreq = 0
ignore_data = 0
cdu_diu_rreq = 1
Reset
go == 1 &
done_band == 0
cdu_diu_rreq = 0
ignore_data = 0
go == 0
cdu_diu_rreq = 1
ignore_data = 0
Idle

FIG. 1

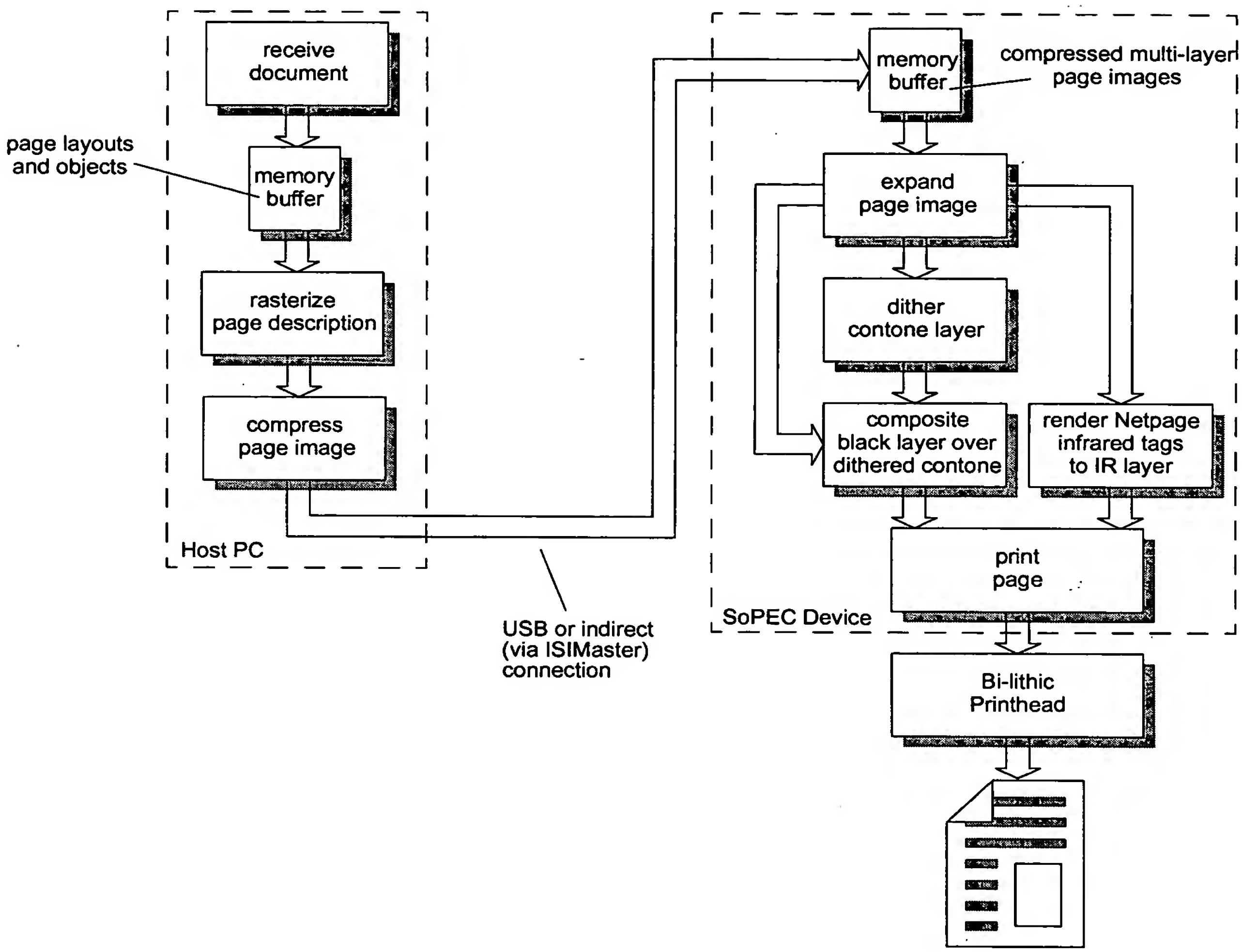

receive document
page layouts and objects
memory buffer
rasterize page description
compress page image
Host PC
memory buffer
compressed multi-layer page images
expand page image
dither contone layer
composite black layer over dithered contone
render Netpage infrared tags to IR layer
print page
SoPEC Device
USB or indirect (via ISIMaster) connection
Bi-lithic Printhead

FIG. 2

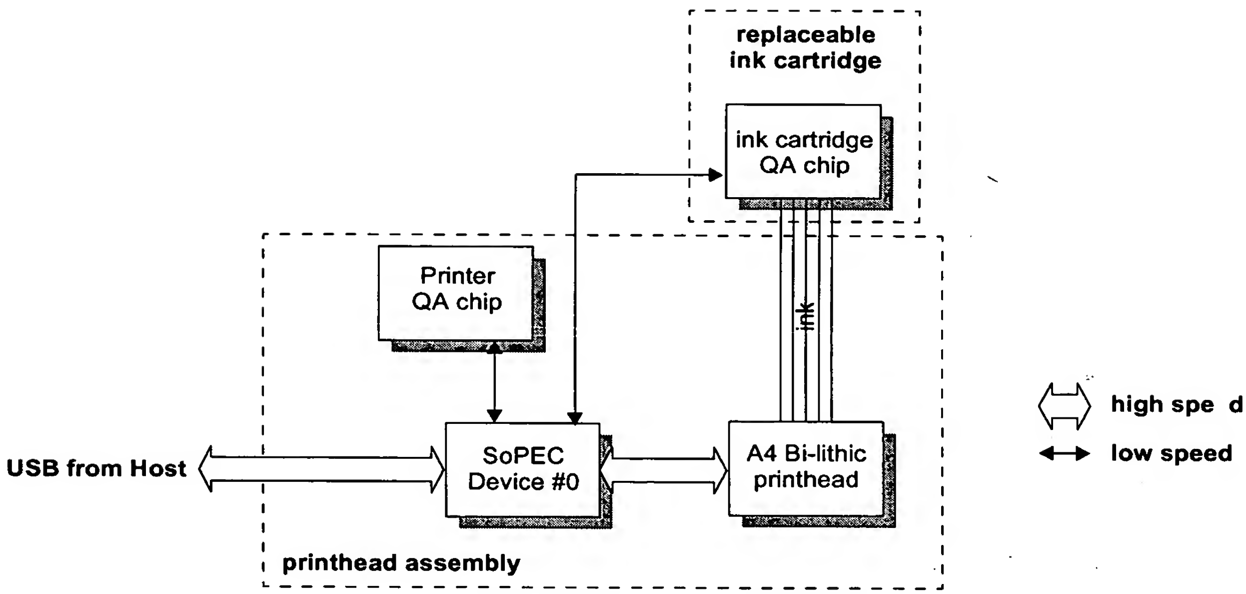

replaceable
ink cartridge
ink cartridge
QA chip
Printer
QA chip
ink
USB from Host
SoPEC
Device #0
A4 Bi-lithic
printhead
printhead assembly
high spe d
low speed

FIG. 3

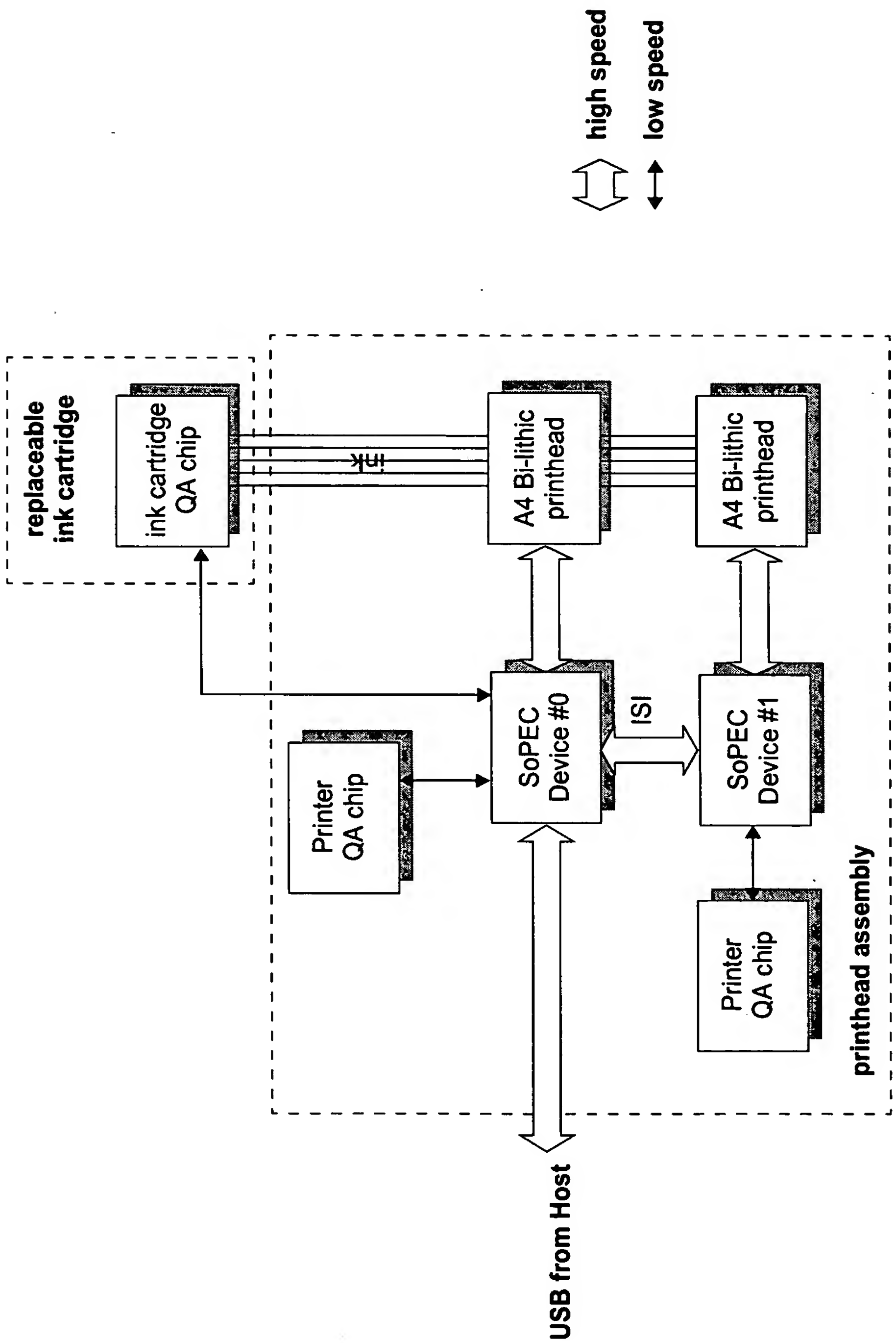

high speed
low speed
replaceable ink cartridge
ink cartridge QA chip
ink
A4 Bi-lithic printhead
A4 Bi-lithic printhead
Printer QA chip
SoPEC Device #0
ISI
SoPEC Device #1
Printer QA chip
printhead assembly
USB from Host

FIG. 4

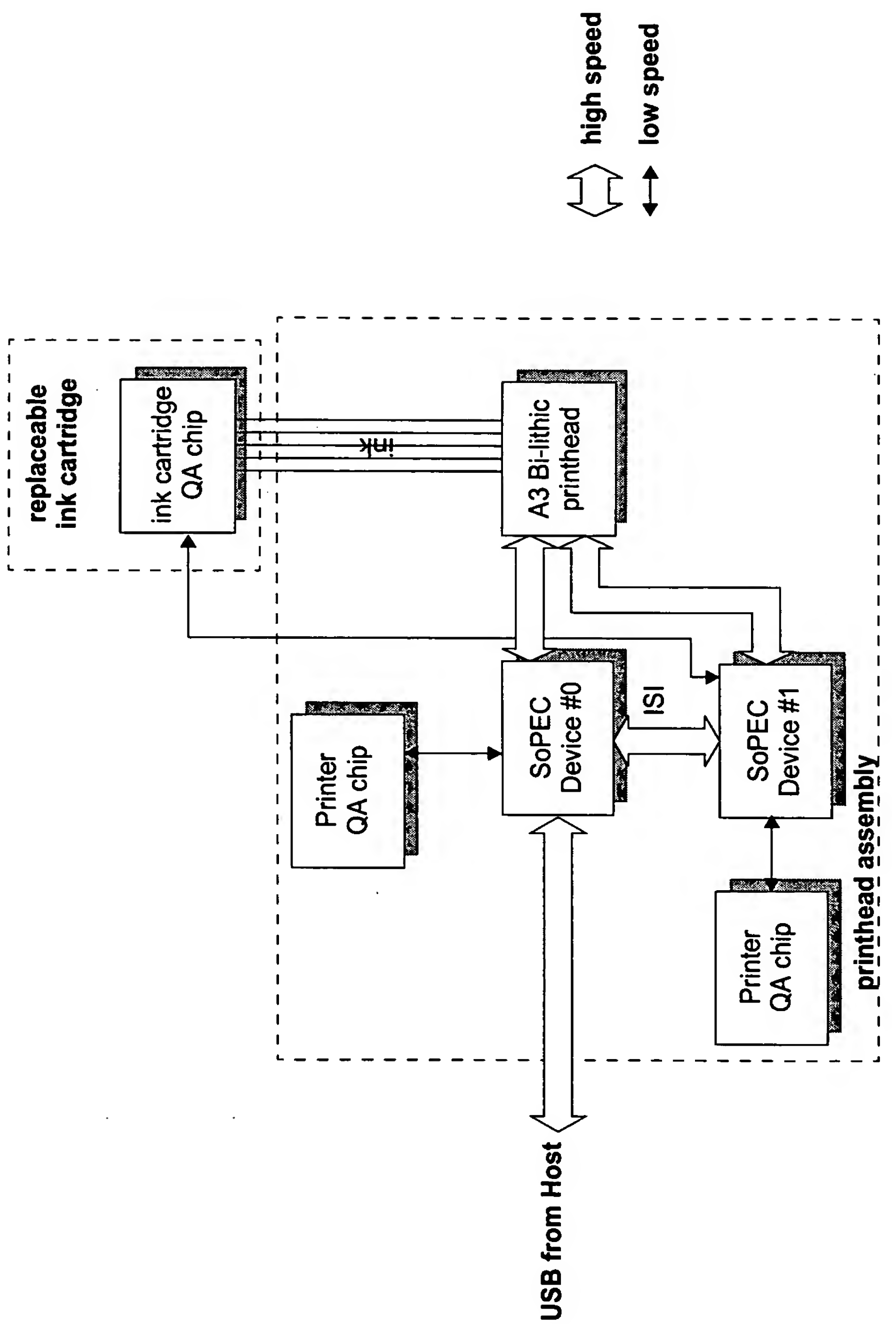

high speed
low speed
replaceable
ink cartridge
ink cartridge
QA chip
ink
A3 Bi-lithic
printhead
Printer
QA chip
SoPEC
Device #0
ISI
SoPEC
Device #1
Printer
QA chip
printhead assembly
USB from Host

FIG. 5

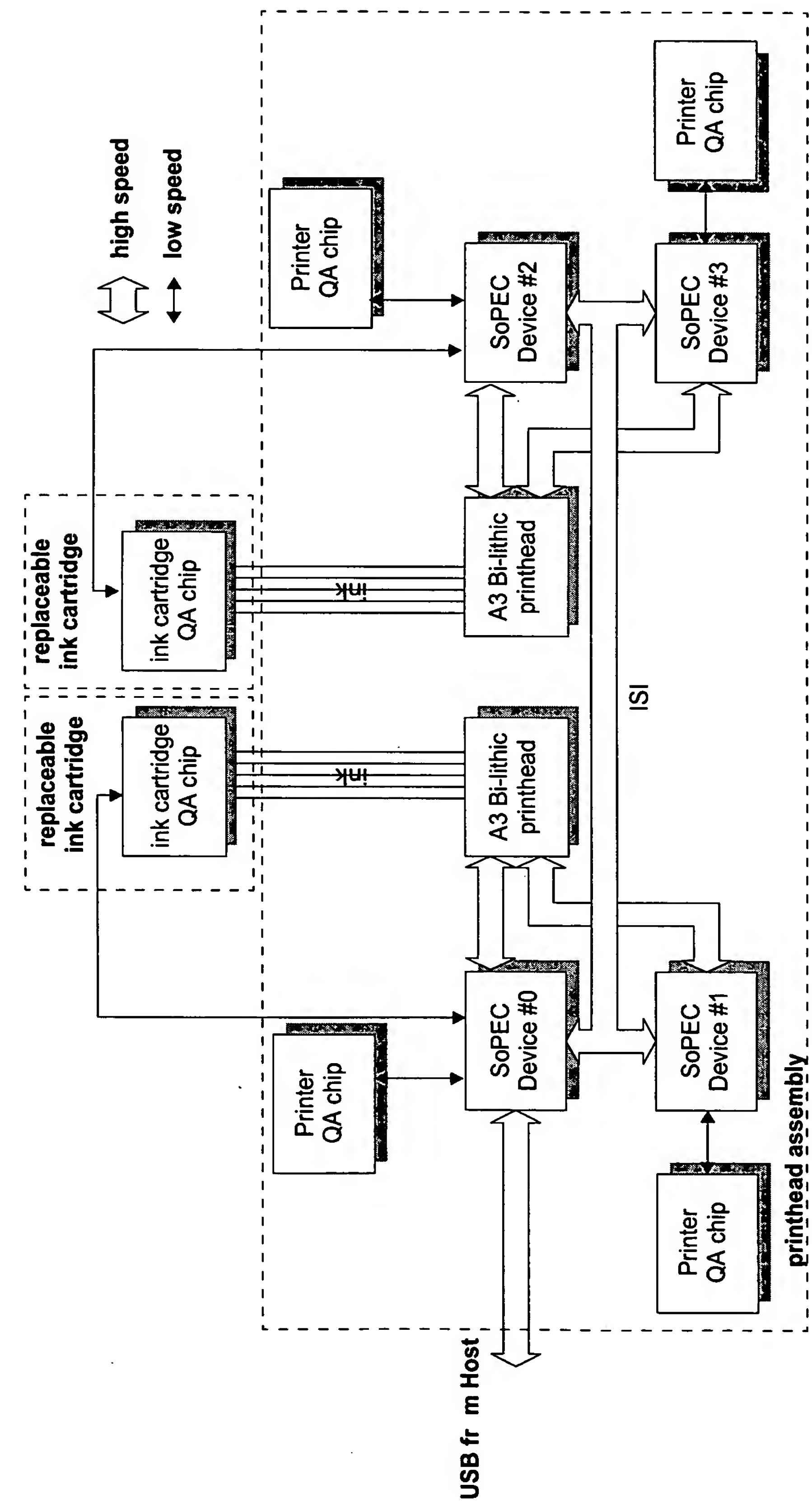

high speed
low speed
replaceable ink cartridge
ink cartridge QA chip
replaceable ink cartridge
ink cartridge QA chip
Printer QA chip
SoPEC Device #2
A3 Bi-lithic printhead
ink
A3 Bi-lithic printhead
ink
SoPEC Device #0
Printer QA chip
Printer QA chip
SoPEC Device #3
ISI
SoPEC Device #1
Printer QA chip
printhead assembly
USB fr m Host

FIG. 6

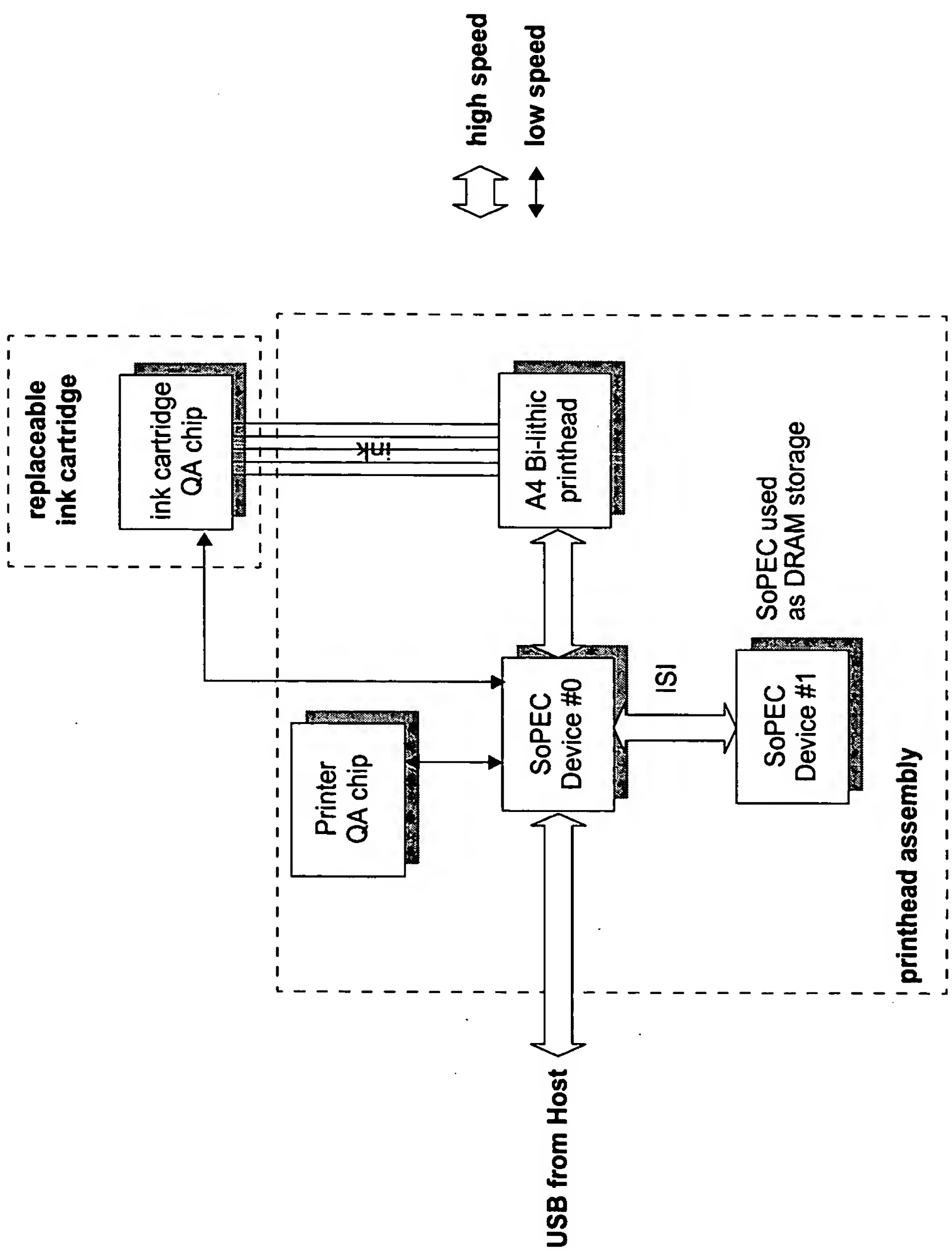

high speed
low speed
replaceable
ink cartridge
ink cartridge
QA chip
ink
A4 Bi-lithic
printhead
SoPEC used
as DRAM storage
Printer
QA chip
SoPEC
Device #0
ISI
SoPEC
Device #1
printhead assembly
USB from Host

FIG. 7

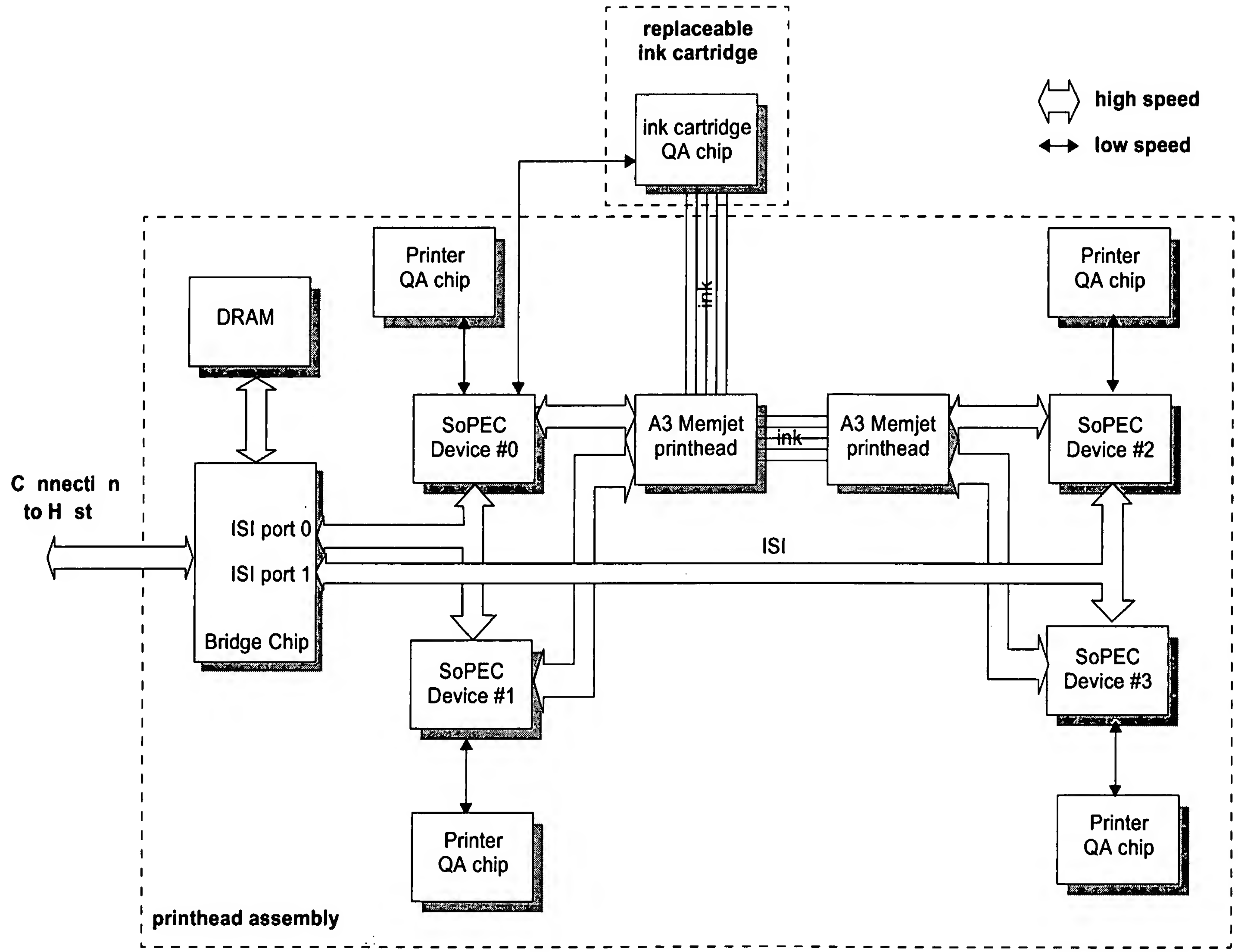

replaceable
ink cartridge
ink cartridge
QA chip
high speed
low speed
Printer
QA chip
DRAM
SoPEC
Device #0
ink
A3 Memjet
printhead
A3 Memjet
printhead
Printer
QA chip
SoPEC
Device #2
C nnecti n
to H st
ISI port 0
ISI port 1
Bridge Chip
ISI
SoPEC
Device #1
SoPEC
Device #3
Printer
QA chip
Printer
QA chip
printhead assembly

FIG. 8

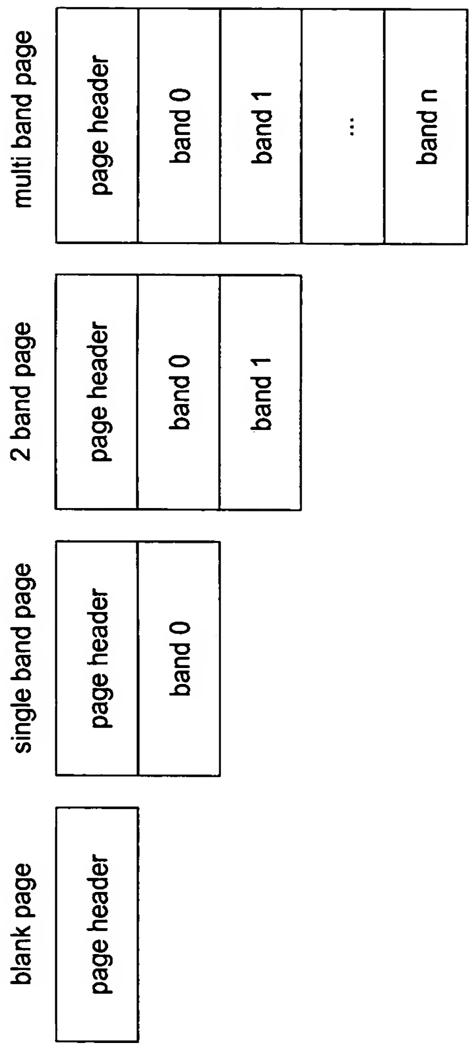

blank page
page header
single band page
page header
band 0
2 band page
page header
band 0
band 1
multi band page
page header
band 0
band 1
...
band n

FIG. 9

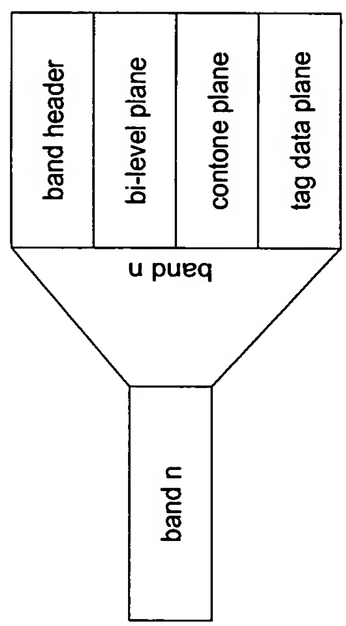

band n
band n
band header
bi-level plane
contone plane
tag data plane

FIG. 10

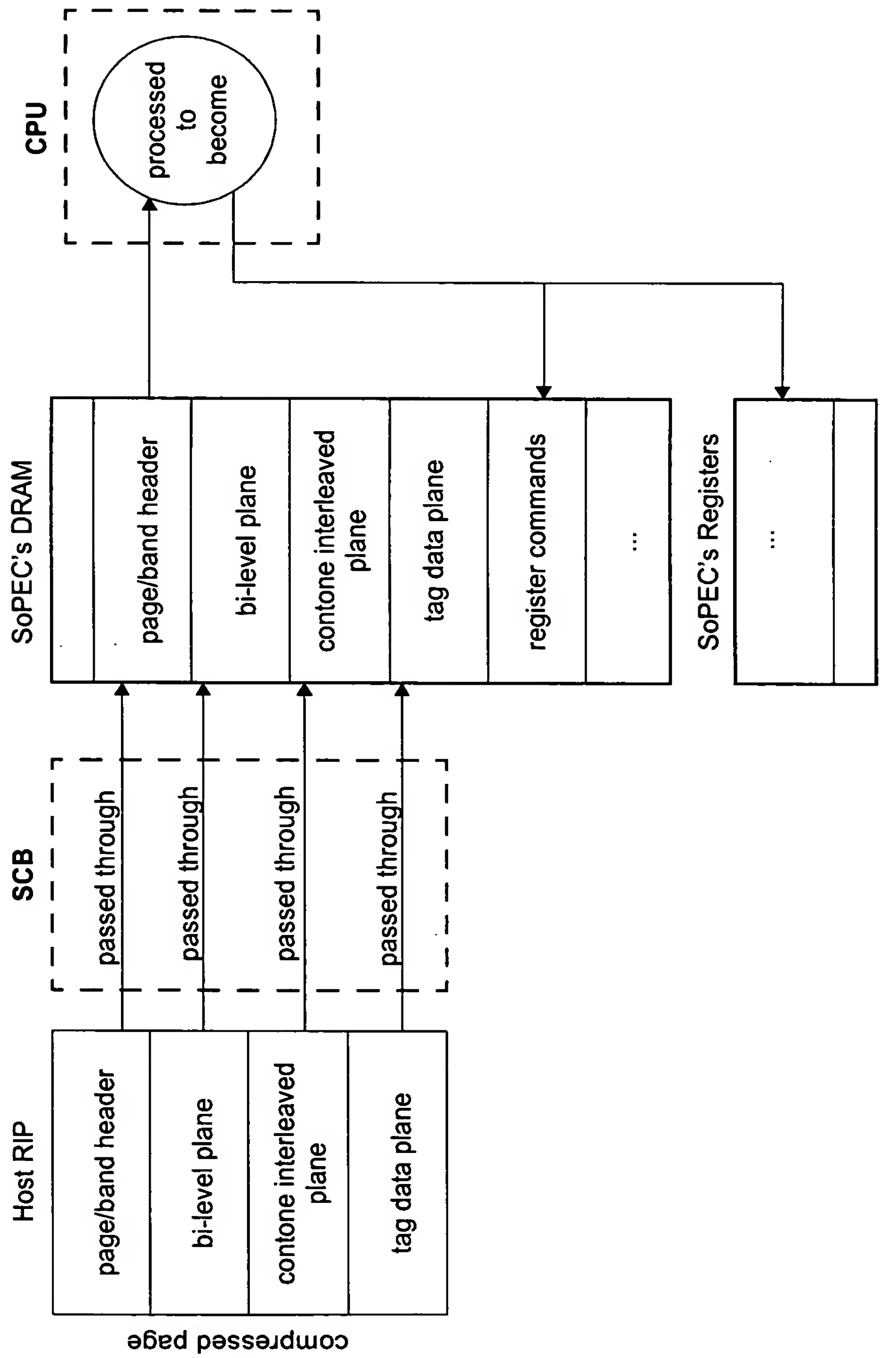

CPU
processed to become
SoPEC's DRAM
page/band header
bi-level plane
contone interleaved plane
tag data plane
register commands
...
SoPEC's Registers
...
SCB
passed through
passed through
passed through
passed through
Host RIP
page/band header
bi-level plane
contone interleaved plane
tag data plane
compressed page

FIG. 11

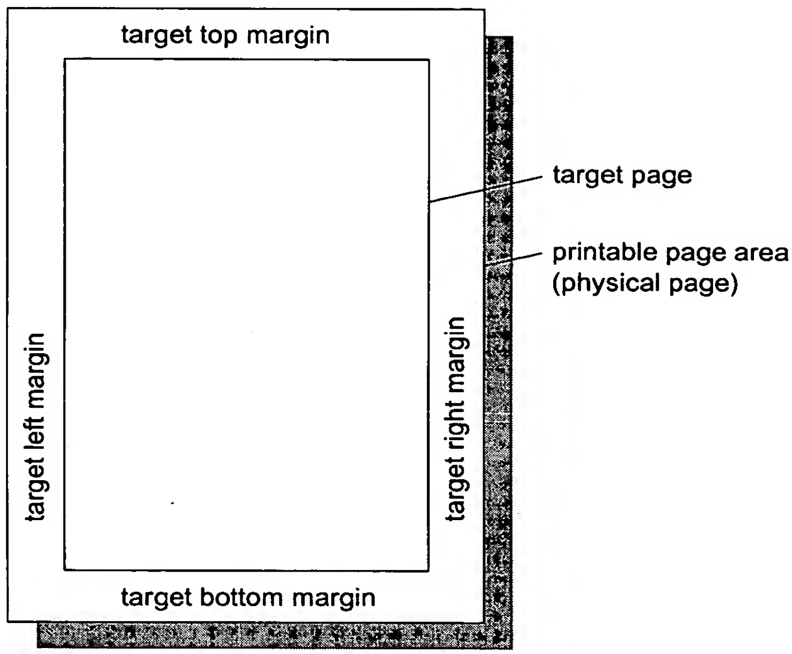
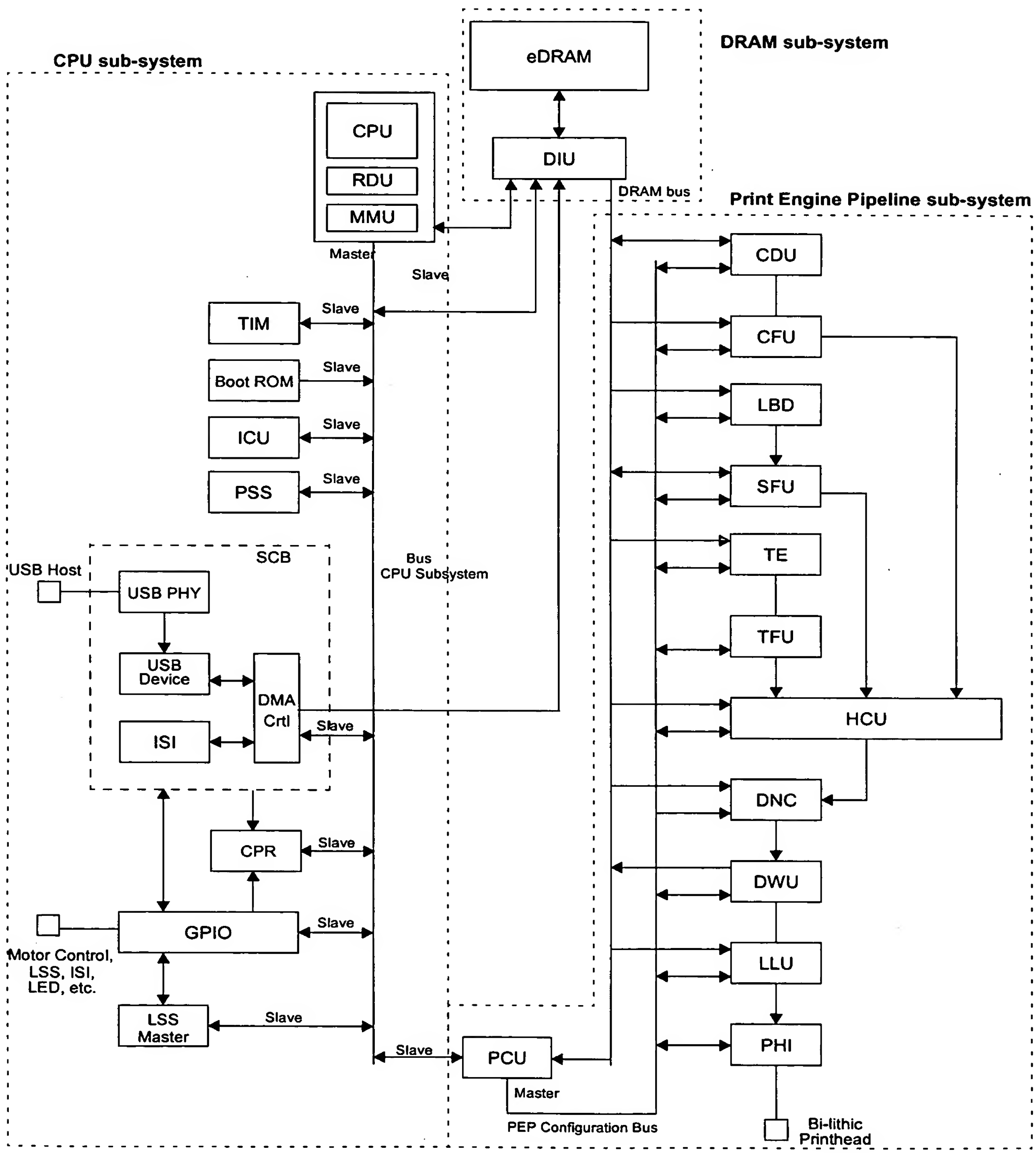

target top margin
target left margin
target right margin
target bottom margin
target page
printable page area
(physical page)

FIG. 12



CPU sub-system
DRAM sub-system
eDRAM
DIU
DRAM bus
Print Engine Pipeline sub-system
CPU
RDU
MMU
Master
Slave
TIM
Slave
Boot ROM
Slave
ICU
Slave
PSS
Slave
SCB
USB Host
USB PHY
USB Device
DMA Crtl
ISI
Slave
Bus CPU Subsystem
CPR
Slave
GPIO
Slave
Motor Control, LSS, ISI, LED, etc.
LSS Master
Slave
Slave
PCU
Master
PEP Configuration Bus
CDU
CFU
LBD
SFU
TE
TFU
HCU
DNC
DWU
LLU
PHI
Bi-lithic Printhead

FIG. 13

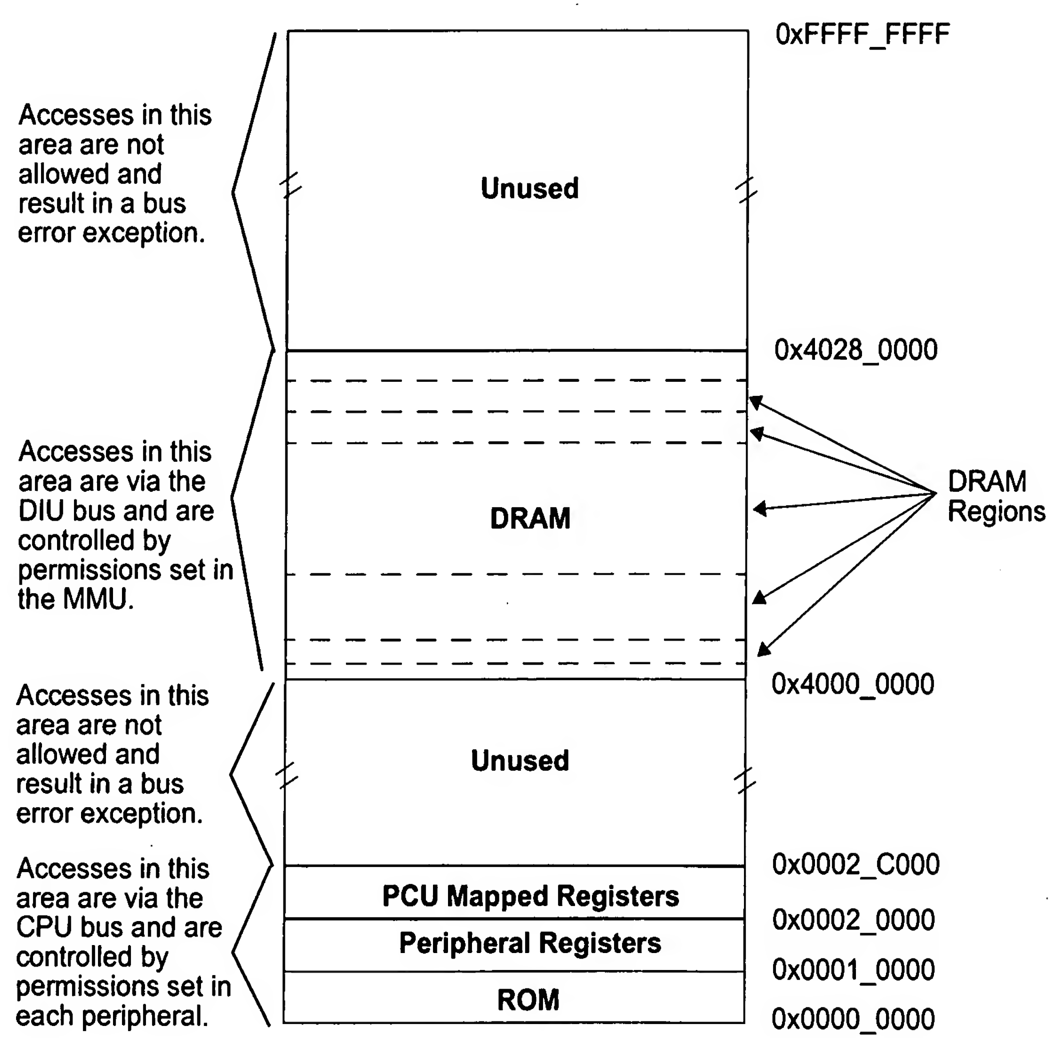

0xFFFF_FFFF
Accesses in this area are not allowed and result in a bus error exception.
Unused
0x4028_0000
Accesses in this area are via the DIU bus and are controlled by permissions set in the MMU.
DRAM
DRAM Regions
0x4000_0000
Accesses in this area are not allowed and result in a bus error exception.
Unused
0x0002_C000
Accesses in this area are via the CPU bus and are controlled by permissions set in each peripheral.
PCU Mapped Registers
0x0002_0000
Peripheral Registers
0x0001_0000
ROM
0x0000_0000

FIG. 14

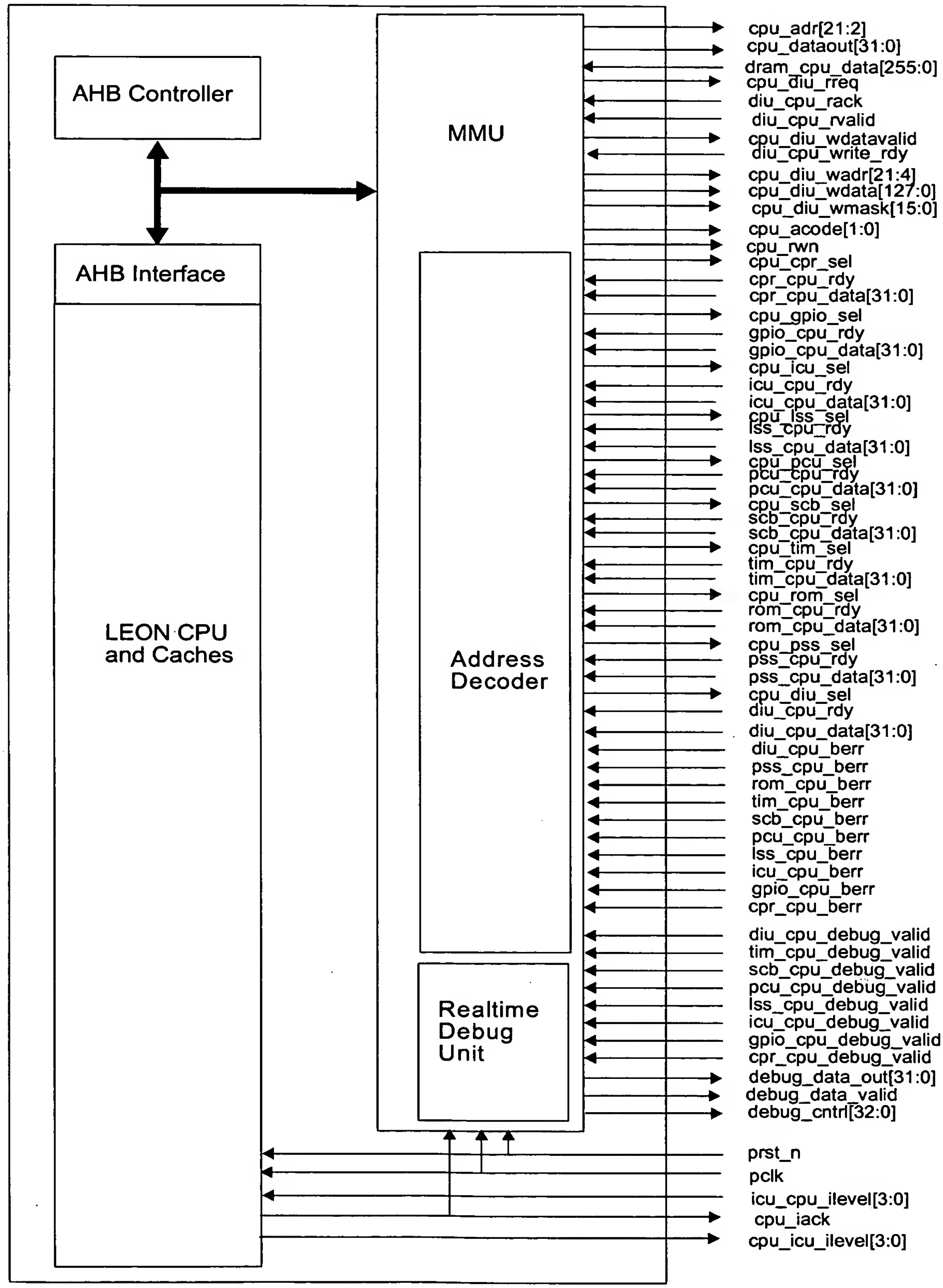

AHB Controller
AHB Interface
LEON CPU and Caches
MMU
Address Decoder
Realtime Debug Unit
cpu_adr[21:2]
cpu_dataout[31:0]
dram_cpu_data[255:0]
cpu_diu_rreq
diu_cpu_rack
diu_cpu_rvalid
cpu_diu_wdatavalid
diu_cpu_write_rdy
cpu_diu_wadr[21:4]
cpu_diu_wdata[127:0]
cpu_diu_wmask[15:0]
cpu_acode[1:0]
cpu_rwn
cpu_cpr_sel
cpr_cpu_rdy
cpr_cpu_data[31:0]
cpu_gpio_sel
gpio_cpu_rdy
gpio_cpu_data[31:0]
cpu_icu_sel
icu_cpu_rdy
icu_cpu_data[31:0]
cpu_lss_sel
lss_cpu_rdy
lss_cpu_data[31:0]
cpu_pcu_sel
pcu_cpu_rdy
pcu_cpu_data[31:0]
cpu_scb_sel
scb_cpu_rdy
scb_cpu_data[31:0]
cpu_tim_sel
tim_cpu_rdy
tim_cpu_data[31:0]
cpu_rom_sel
rom_cpu_rdy
rom_cpu_data[31:0]
cpu_pss_sel
pss_cpu_rdy
pss_cpu_data[31:0]
cpu_diu_sel
diu_cpu_rdy
diu_cpu_data[31:0]
diu_cpu_berr
pss_cpu_berr
rom_cpu_berr
tim_cpu_berr
scb_cpu_berr
pcu_cpu_berr
lss_cpu_berr
icu_cpu_berr
gpio_cpu_berr
cpr_cpu_berr
diu_cpu_debug_valid
tim_cpu_debug_valid
scb_cpu_debug_valid
pcu_cpu_debug_valid
lss_cpu_debug_valid
icu_cpu_debug_valid
gpio_cpu_debug_valid
cpr_cpu_debug_valid
debug_data_out[31:0]
debug_data_valid
debug_cntrl[32:0]
prst_n
pclk
icu_cpu_ilevel[3:0]
cpu_iack
cpu_icu_ilevel[3:0]

FIG. 15

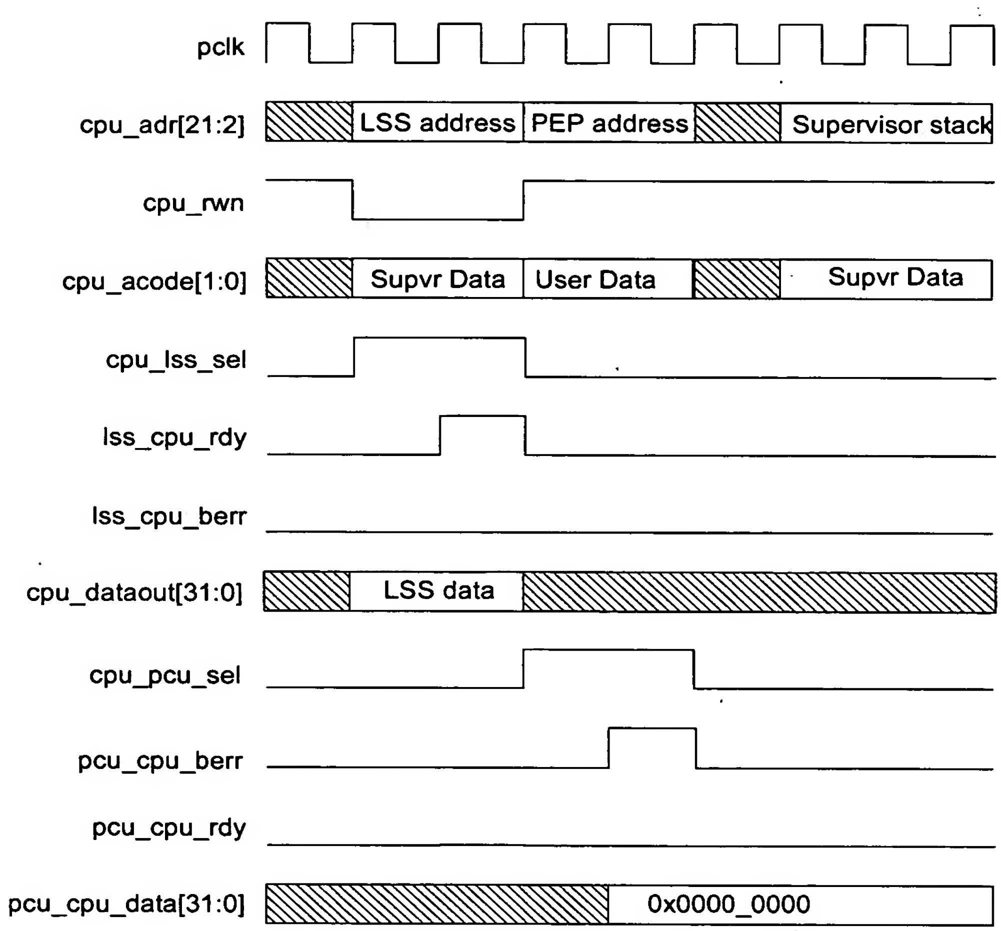

pclk
cpu_adr[21:2]
LSS address
PEP address
Supervisor stack
cpu_rwn
cpu_acode[1:0]
Supvr Data
User Data
Supvr Data
cpu_lss_sel
lss_cpu_rdy
lss_cpu_berr
cpu_dataout[31:0]
LSS data
cpu_pcu_sel
pcu_cpu_berr
pcu_cpu_rdy
pcu_cpu_data[31:0]
0x0000_0000

FIG. 16

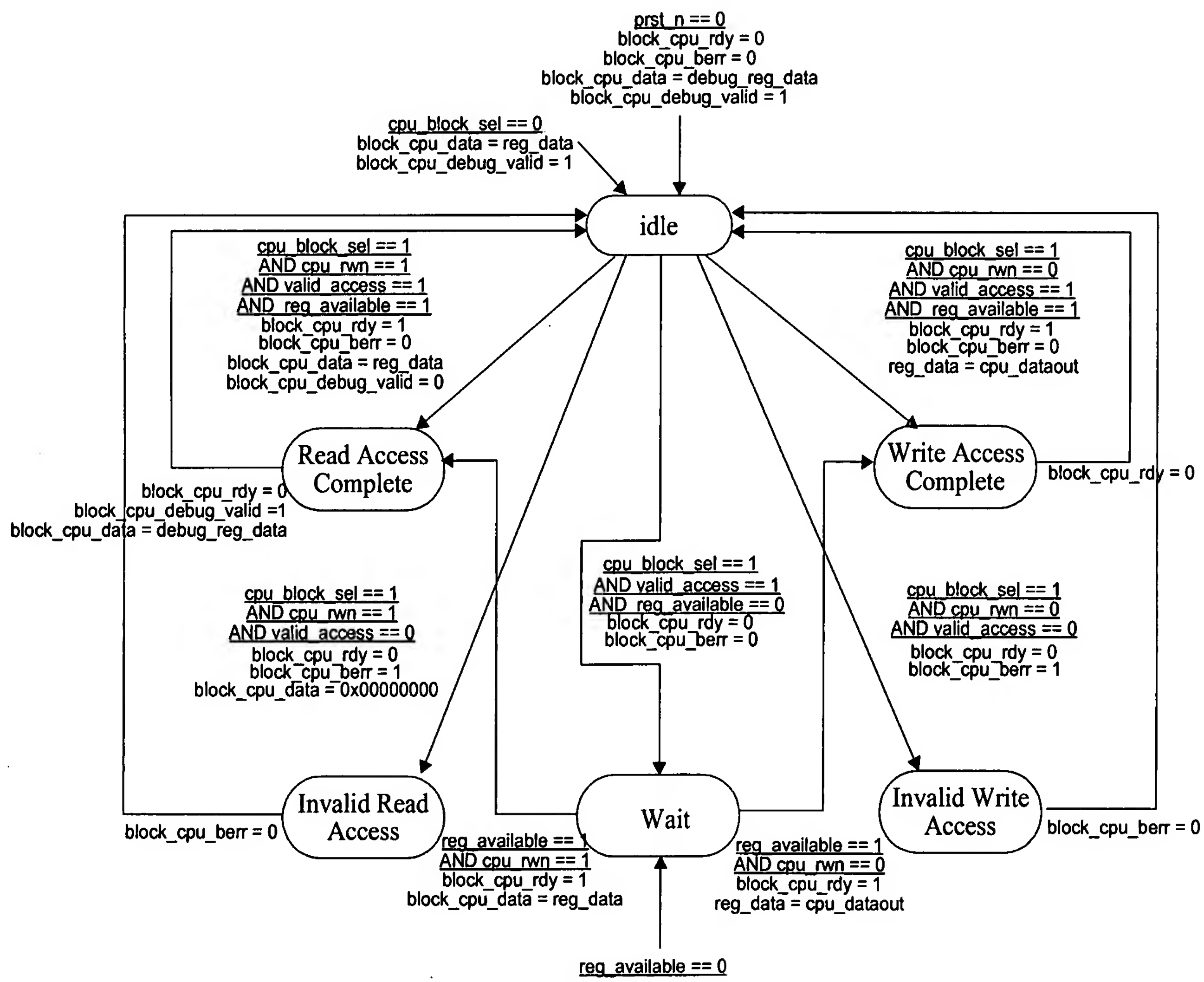

prst_n == 0
block_cpu_rdy = 0
block_cpu_berr = 0
block_cpu_data = debug_reg_data
block_cpu_debug_valid = 1
cpu_block_sel == 0
block_cpu_data = reg_data
block_cpu_debug_valid = 1
idle
cpu_block_sel == 1
AND cpu_rwn == 1
AND valid_access == 1
AND reg_available == 1
block_cpu_rdy = 1
block_cpu_berr = 0
block_cpu_data = reg_data
block_cpu_debug_valid = 0
cpu_block_sel == 1
AND cpu_rwn == 0
AND valid_access == 1
AND reg_available == 1
block_cpu_rdy = 1
block_cpu_berr = 0
reg_data = cpu_dataout
Read Access Complete
Write Access Complete
block_cpu_rdy = 0
block_cpu_debug_valid =1
block_cpu_data = debug_reg_data
block_cpu_rdy = 0
cpu_block_sel == 1
AND valid_access == 1
AND reg_available == 0
block_cpu_rdy = 0
block_cpu_berr = 0
cpu_block_sel == 1
AND cpu_rwn == 1
AND valid_access == 0
block_cpu_rdy = 0
block_cpu_berr = 1
block_cpu_data = 0x00000000
cpu_block_sel == 1
AND cpu_rwn == 0
AND valid_access == 0
block_cpu_rdy = 0
block_cpu_berr = 1
Invalid Read Access
Wait
Invalid Write Access
block_cpu_berr = 0
block_cpu_berr = 0
reg_available == 1
AND cpu_rwn == 1
block_cpu_rdy = 1
block_cpu_data = reg_data
reg_available == 1
AND cpu_rwn == 0
block_cpu_rdy = 1
reg_data = cpu_dataout
reg_available == 0

FIG. 17

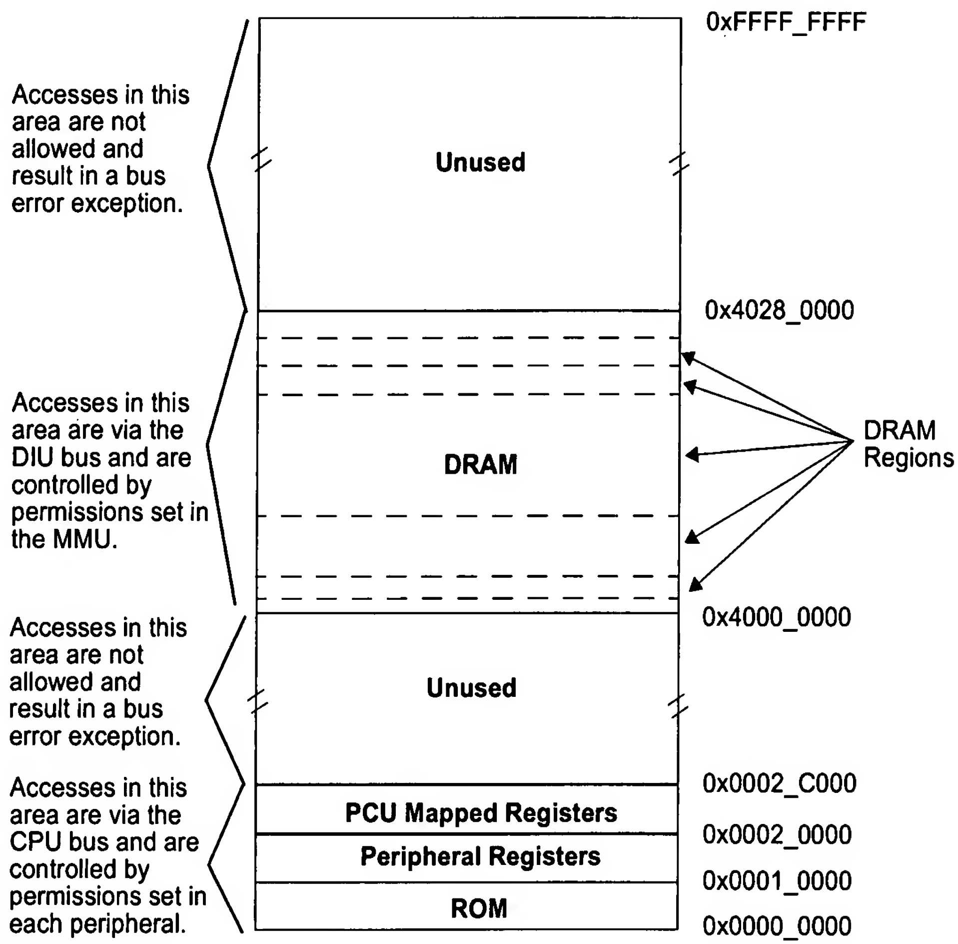

0xFFFF_FFFF
Accesses in this area are not allowed and result in a bus error exception.
Unused
0x4028_0000
Accesses in this area are via the DIU bus and are controlled by permissions set in the MMU.
DRAM
DRAM Regions
0x4000_0000
Accesses in this area are not allowed and result in a bus error exception.
Unused
0x0002_C000
Accesses in this area are via the CPU bus and are controlled by permissions set in each peripheral.
PCU Mapped Registers
0x0002_0000
Peripheral Registers
0x0001_0000
ROM
0x0000_0000

FIG. 18

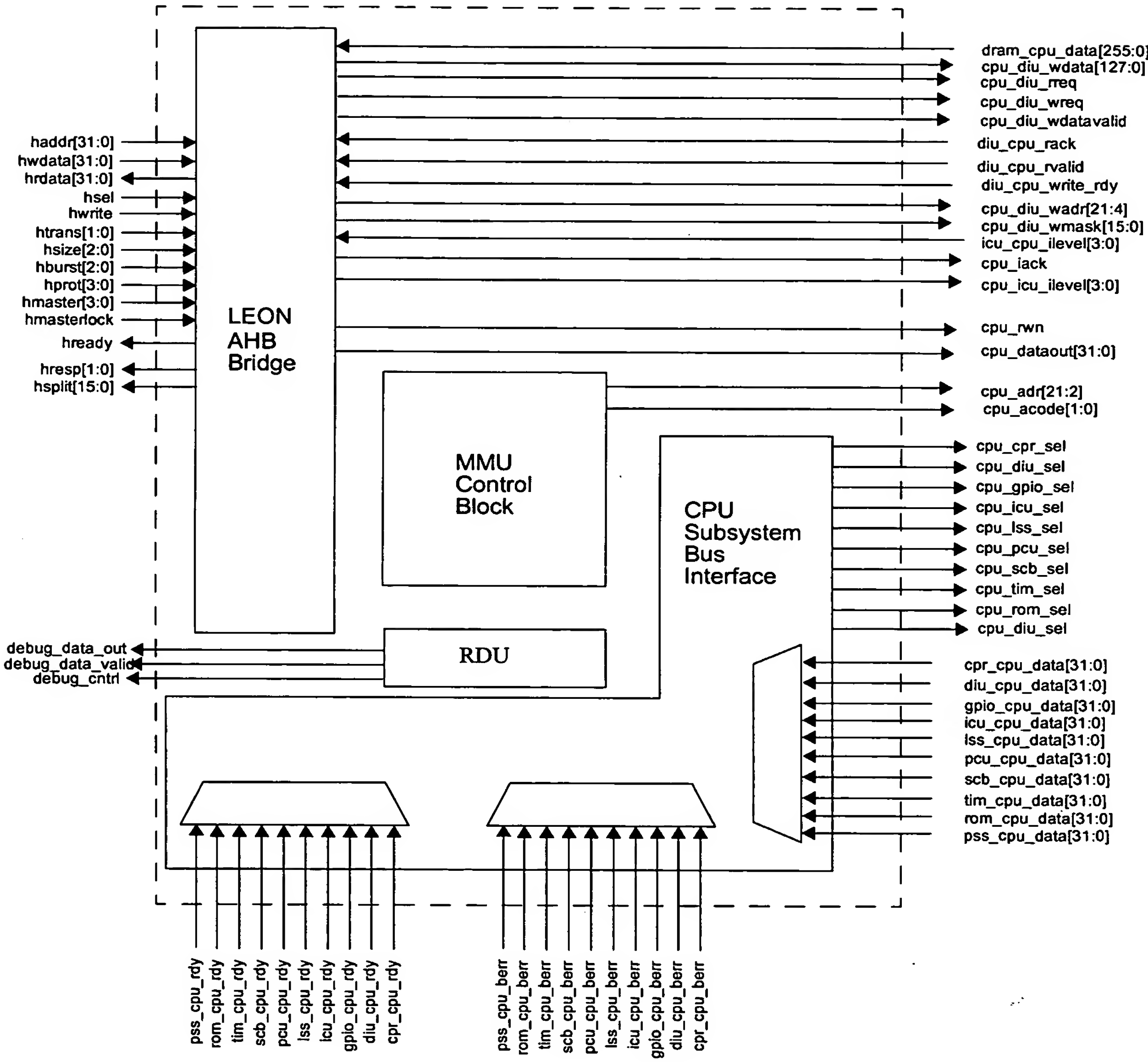

LEON
AHB
Bridge
MMU
Control
Block
CPU
Subsystem
Bus
Interface
RDU
haddr[31:0]
hwdata[31:0]
hrdata[31:0]
hsel
hwrite
htrans[1:0]
hsize[2:0]
hburst[2:0]
hprot[3:0]
hmaster[3:0]
hmasterlock
hready
hresp[1:0]
hsplit[15:0]
debug_data_out
debug_data_valid
debug_cntrl
dram_cpu_data[255:0]
cpu_diu_wdata[127:0]
cpu_diu_rreq
cpu_diu_wreq
cpu_diu_wdatavalid
diu_cpu_rack
diu_cpu_rvalid
diu_cpu_write_rdy
cpu_diu_wadr[21:4]
cpu_diu_wmask[15:0]
icu_cpu_ilevel[3:0]
cpu_iack
cpu_icu_ilevel[3:0]
cpu_rwn
cpu_dataout[31:0]
cpu_adr[21:2]
cpu_acode[1:0]
cpu_cpr_sel
cpu_diu_sel
cpu_gpio_sel
cpu_icu_sel
cpu_lss_sel
cpu_pcu_sel
cpu_scb_sel
cpu_tim_sel
cpu_rom_sel
cpu_diu_sel
cpr_cpu_data[31:0]
diu_cpu_data[31:0]
gpio_cpu_data[31:0]
icu_cpu_data[31:0]
lss_cpu_data[31:0]
pcu_cpu_data[31:0]
scb_cpu_data[31:0]
tim_cpu_data[31:0]
rom_cpu_data[31:0]
pss_cpu_data[31:0]
pss_cpu_rdy
rom_cpu_rdy
tim_cpu_rdy
scb_cpu_rdy
pcu_cpu_rdy
lss_cpu_rdy
icu_cpu_rdy
gpio_cpu_rdy
diu_cpu_rdy
cpr_cpu_rdy
pss_cpu_berr
rom_cpu_berr
tim_cpu_berr
scb_cpu_berr
pcu_cpu_berr
lss_cpu_berr
icu_cpu_berr
gpio_cpu_berr
diu_cpu_berr
cpr_cpu_berr

FIG. 19

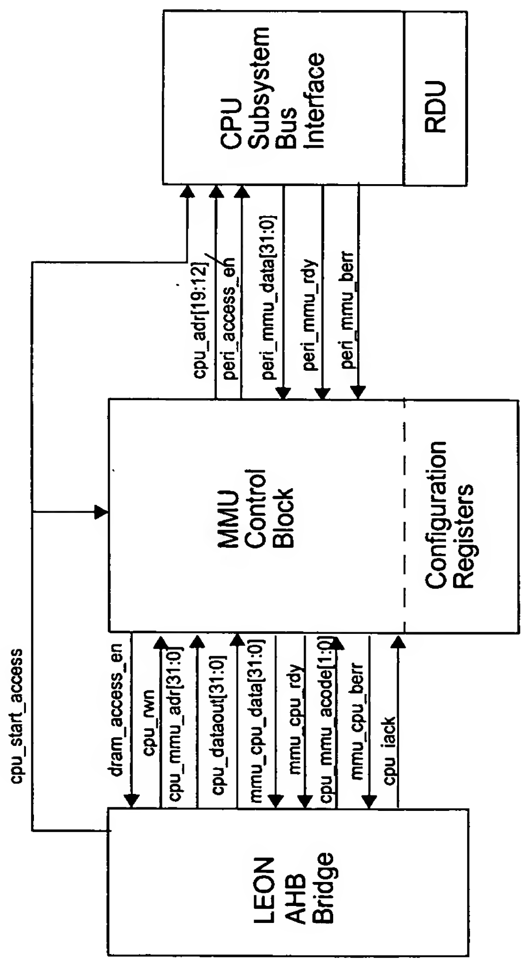

LEON AHB Bridge
MMU Control Block
Configuration Registers
CPU Subsystem Bus Interface
RDU
cpu_start_access
dram_access_en
cpu_rwn
cpu_mmu_adr[31:0]
cpu_dataout[31:0]
mmu_cpu_data[31:0]
mmu_cpu_rdy
cpu_mmu_acode[1:0]
mmu_cpu_berr
cpu_iack
cpu_adr[19:12]
peri_access_en
peri_mmu_data[31:0]
peri_mmu_rdy
peri_mmu_berr

FIG. 20

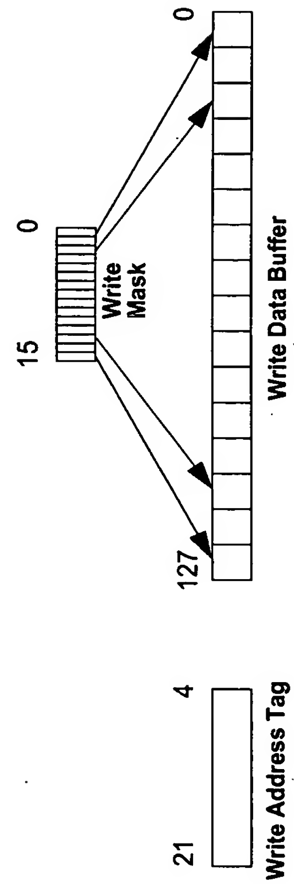

21
4
Write Address Tag
15
0
Write Mask
127
0
Write Data Buffer

FIG. 21

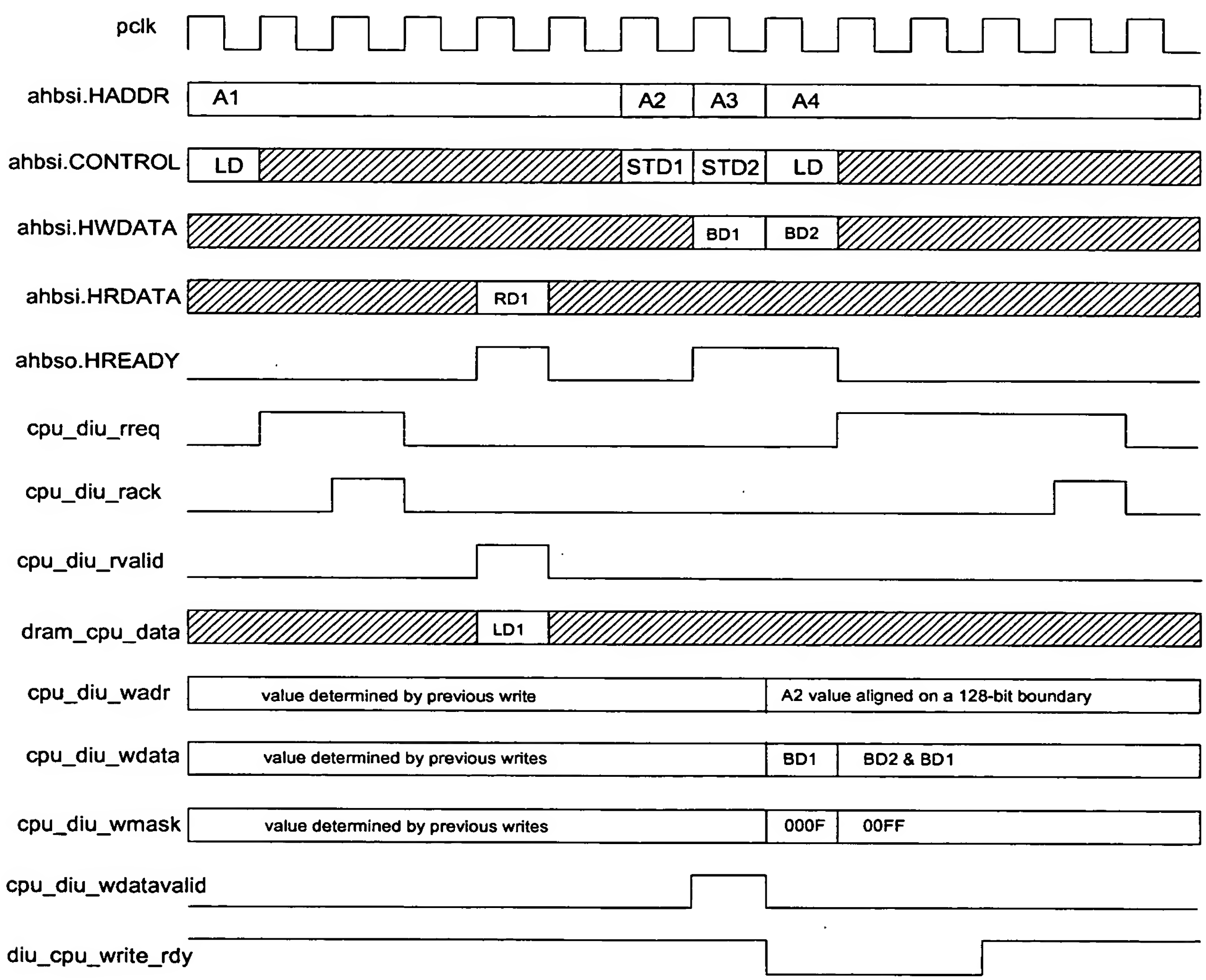

pclk
ahbsi.HADDR
A1
A2
A3
A4
ahbsi.CONTROL
LD
STD1
STD2
LD
ahbsi.HWDATA
BD1
BD2
ahbsi.HRDATA
RD1
ahbso.HREADY
cpu_diu_rreq
cpu_diu_rack
cpu_diu_rvalid
dram_cpu_data
LD1
cpu_diu_wadr
value determined by previous write
A2 value aligned on a 128-bit boundary
cpu_diu_wdata
value determined by previous writes
BD1
BD2 & BD1
cpu_diu_wmask
value determined by previous writes
000F
00FF
cpu_diu_wdatavalid
diu_cpu_write_rdy

FIG. 22

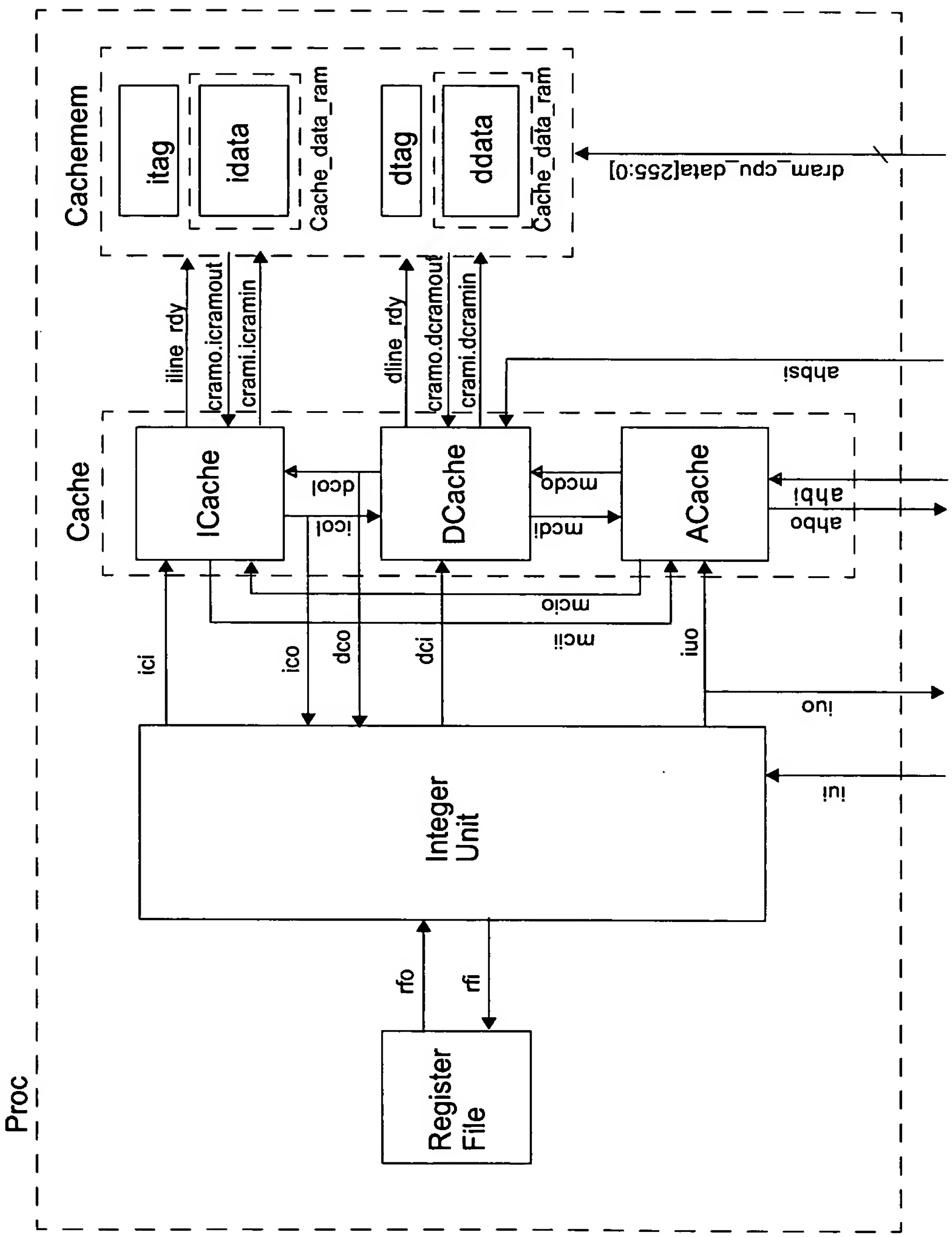
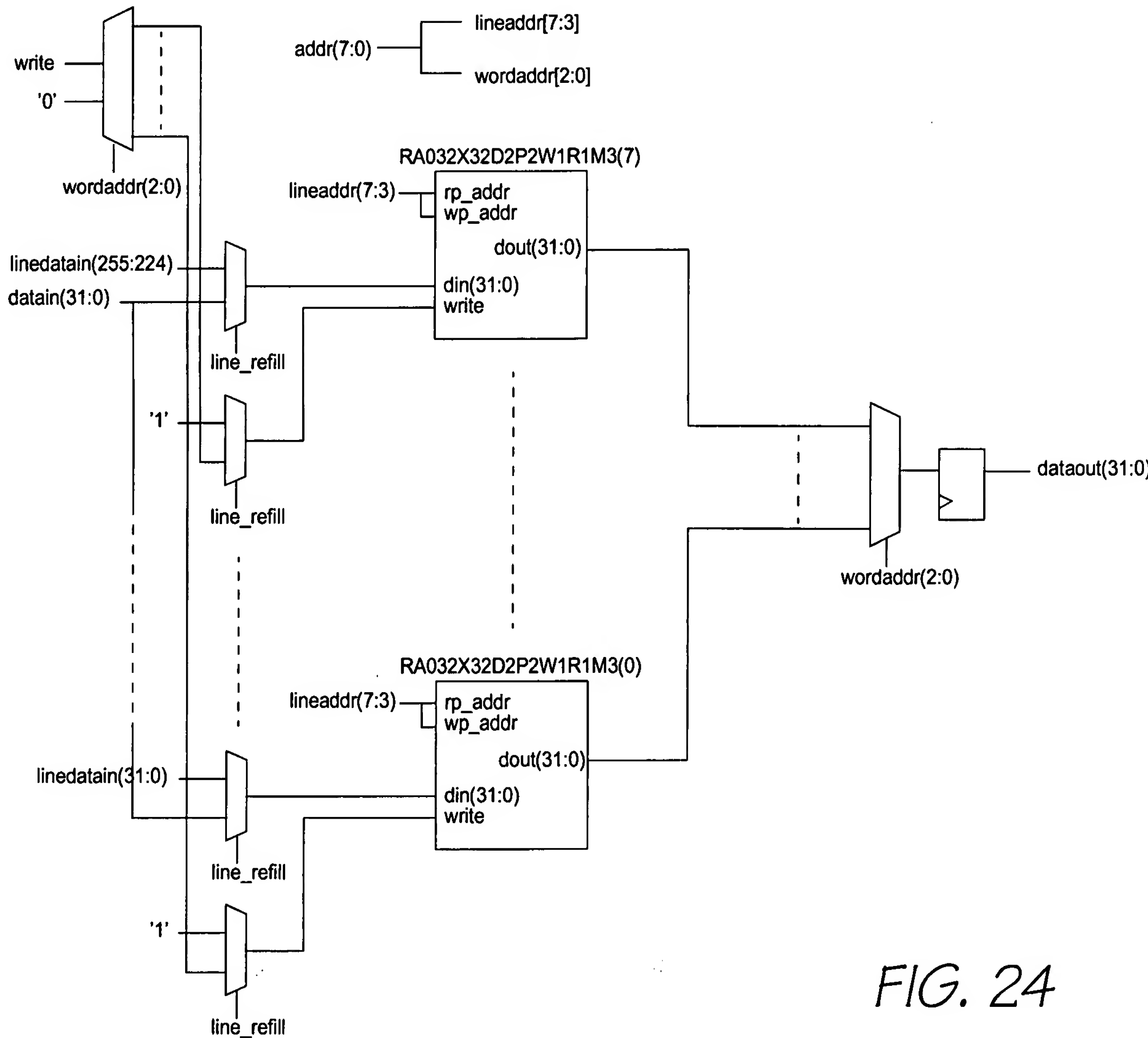

Proc
Cachemem
itag
idata
Cache_data_ram
dtag
ddata
Cache_data_ram
dram_cpu_data[255:0]
iline_rdy
cramo.icramout
crami.icramin
dline_rdy
cramo.dcramout
crami.dcramin
ahbsi
Cache
ICache
DCache
ACache
dcol
icol
mcdo
mcdi
ahbi
ahbo
mcio
mcii
ici
ico
dco
dci
iuo
iui
Integer Unit
rfo
rfi
Register File

FIG. 23



write
'0'
wordaddr(2:0)
addr(7:0)
lineaddr[7:3]
wordaddr[2:0]
RA032X32D2P2W1R1M3(7)
lineaddr(7:3)
rp_addr
wp_addr
dout(31:0)
din(31:0)
write
linedatain(255:224)
datain(31:0)
line_refill
'1'
line_refill
RA032X32D2P2W1R1M3(0)
lineaddr(7:3)
rp_addr
wp_addr
dout(31:0)
din(31:0)
write
linedatain(31:0)
line_refill
'1'
line_refill
wordaddr(2:0)
dataout(31:0)

FIG. 24

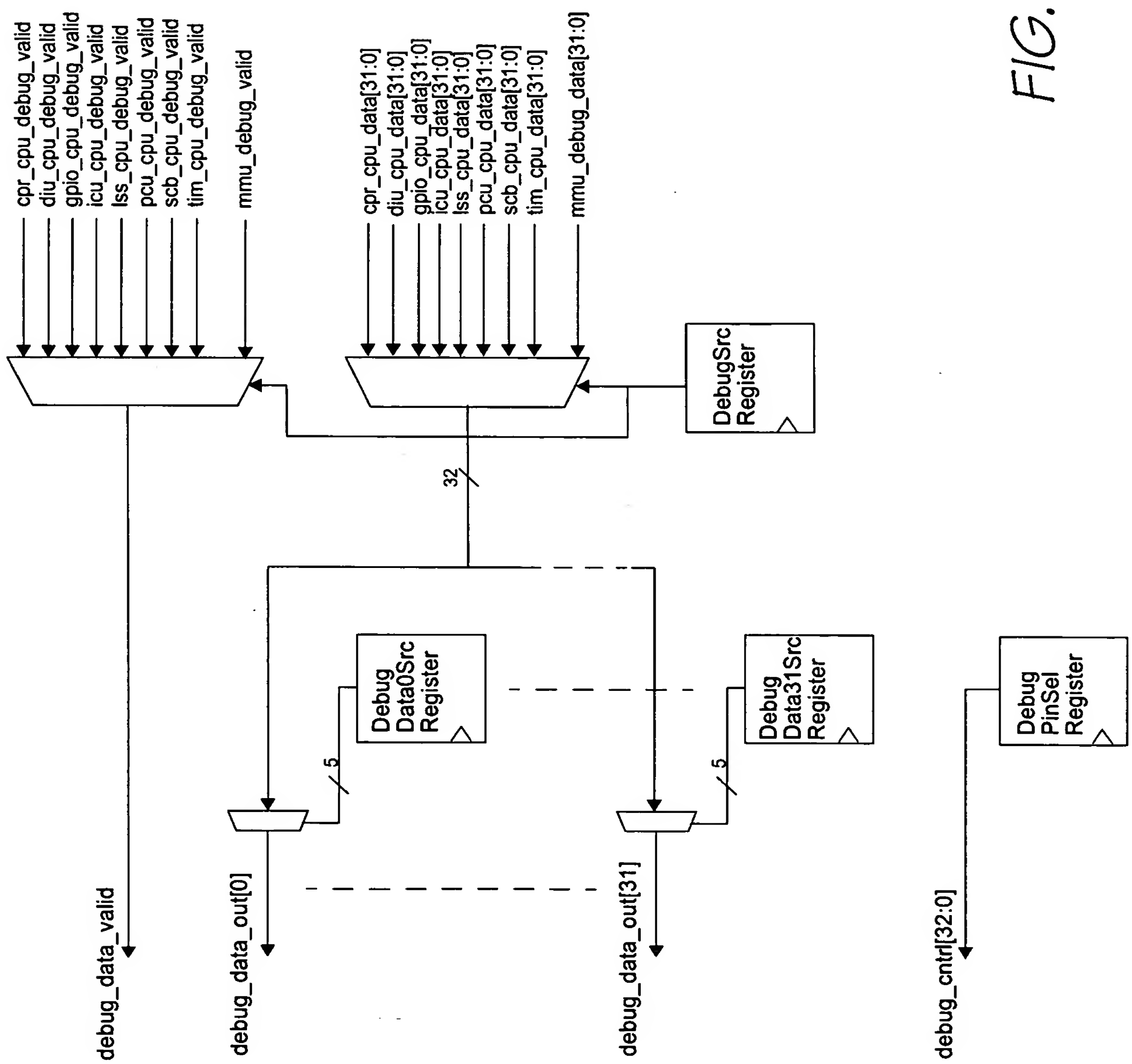

cpr_cpu_debug_valid
diu_cpu_debug_valid
gpio_cpu_debug_valid
icu_cpu_debug_valid
lss_cpu_debug_valid
pcu_cpu_debug_valid
scb_cpu_debug_valid
tim_cpu_debug_valid
mmu_debug_valid
cpr_cpu_data[31:0]
diu_cpu_data[31:0]
gpio_cpu_data[31:0]
icu_cpu_data[31:0]
lss_cpu_data[31:0]
pcu_cpu_data[31:0]
scb_cpu_data[31:0]
tim_cpu_data[31:0]
mmu_debug_data[31:0]
DebugSrc Register
32
Debug Data0Src Register
Debug Data31Src Register
Debug PinSel Register
5
5
debug_data_valid
debug_data_out[0]
debug_data_out[31]
debug_cntrl[32:0]

FIG. 25

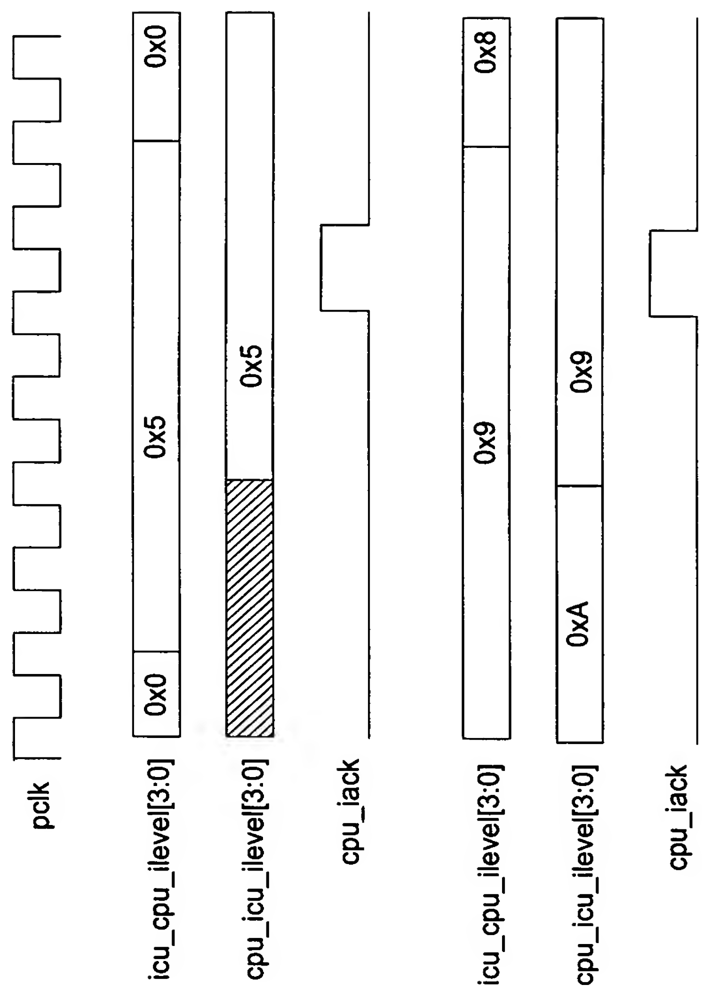

pclk
icu_cpu_ilevel[3:0]
0x0
0x5
0x0
cpu_icu_ilevel[3:0]
0x5
cpu_iack
icu_cpu_ilevel[3:0]
0x9
0x8
cpu_icu_ilevel[3:0]
0xA
0x9
cpu_iack

FIG. 26

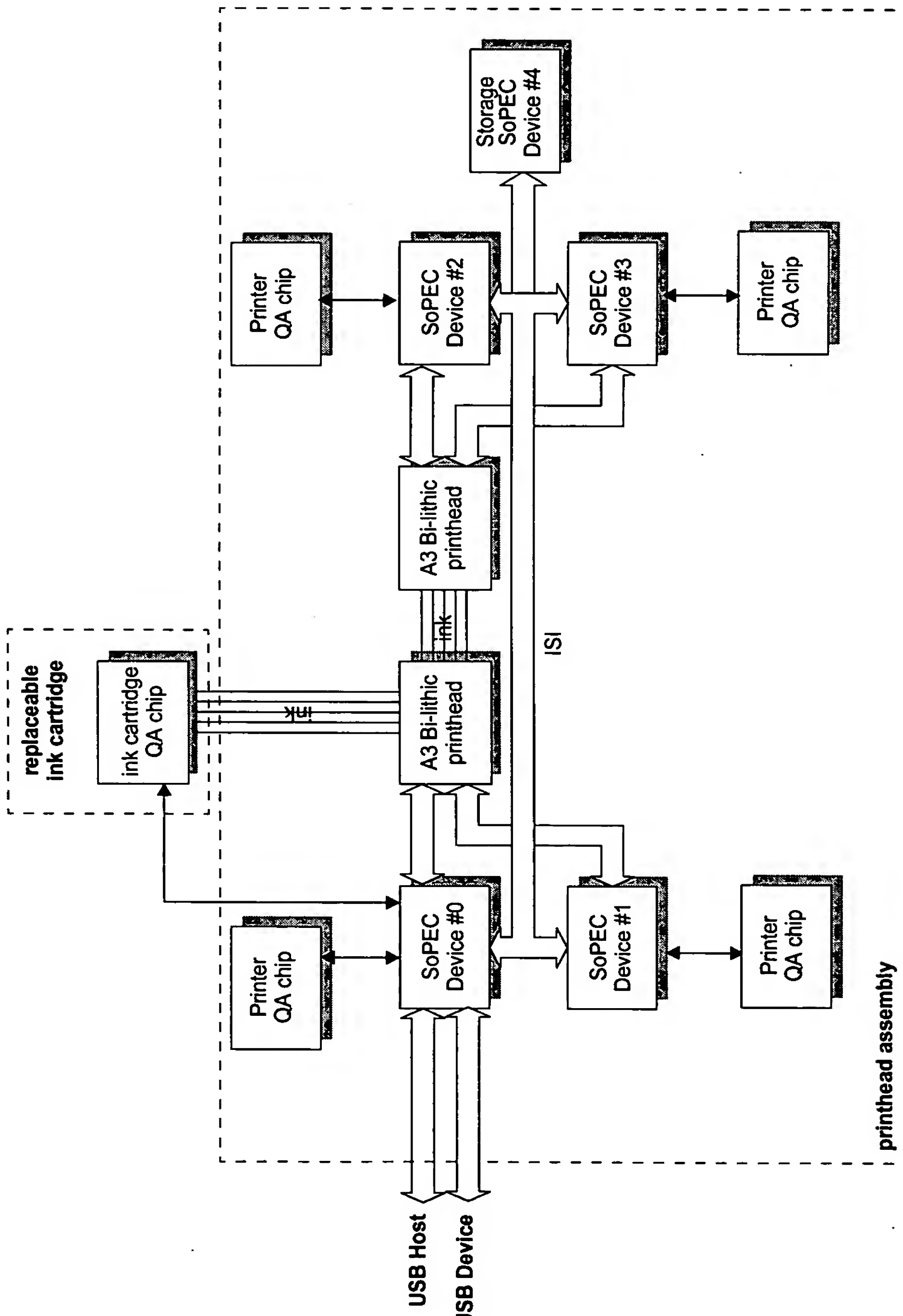

USB Host
USB Device
printhead assembly
replaceable ink cartridge
ink cartridge QA chip
Printer QA chip
SoPEC Device #0
SoPEC Device #1
A3 Bi-lithic printhead
ink
A3 Bi-lithic printhead
ink
ISI
Printer QA chip
SoPEC Device #2
SoPEC Device #3
Printer QA chip
Printer QA chip
Storage SoPEC Device #4

FIG. 27

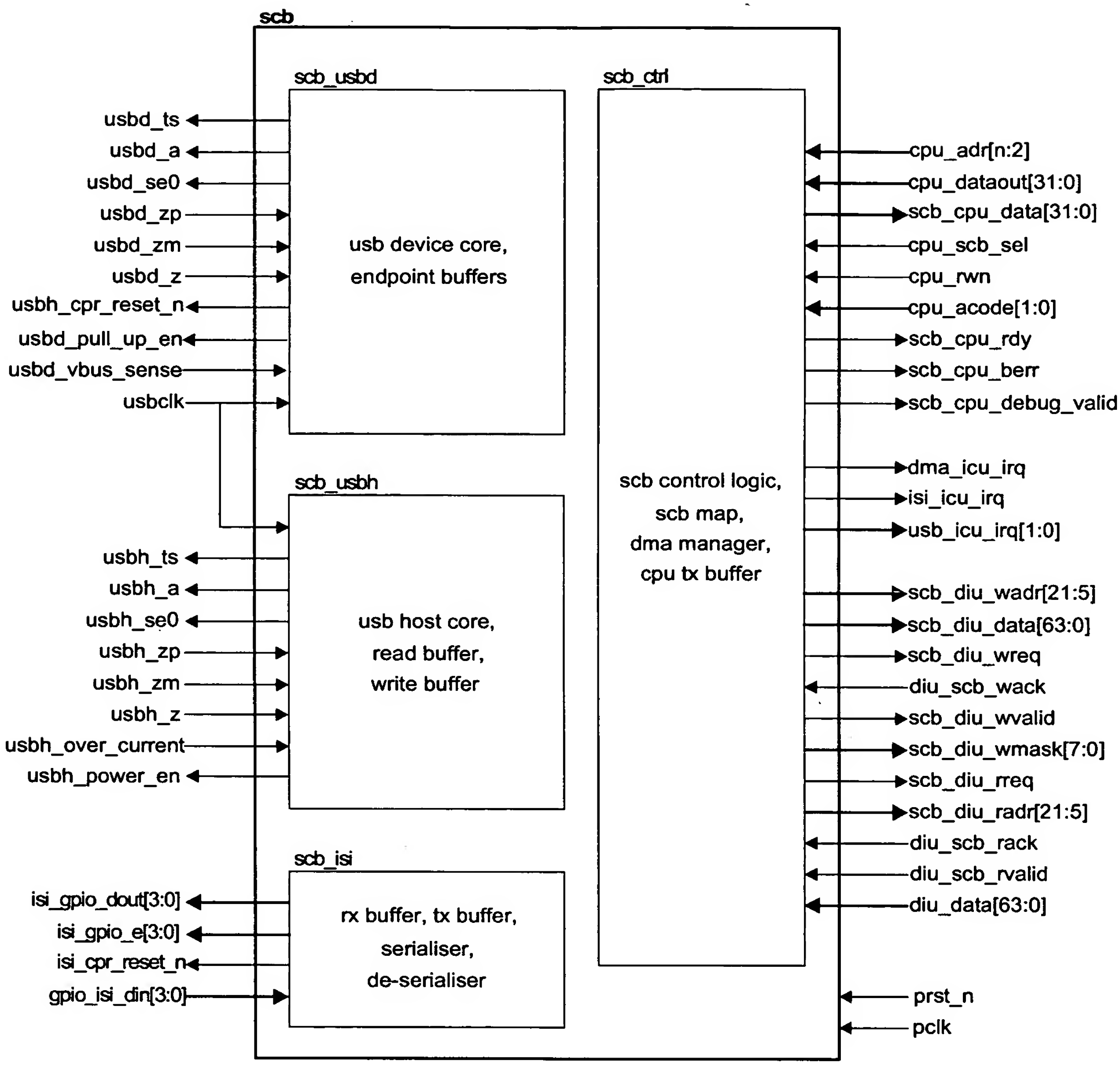

scb
scb_usbd
usb device core,
endpoint buffers
scb_ctrl
scb control logic,
scb map,
dma manager,
cpu tx buffer
scb_usbh
usb host core,
read buffer,
write buffer
scb_isi
rx buffer, tx buffer,
serialiser,
de-serialiser
usbd_ts
usbd_a
usbd_se0
usbd_zp
usbd_zm
usbd_z
usbh_cpr_reset_n
usbd_pull_up_en
usbd_vbus_sense
usbclk
usbh_ts
usbh_a
usbh_se0
usbh_zp
usbh_zm
usbh_z
usbh_over_current
usbh_power_en
isi_gpio_dout[3:0]
isi_gpio_e[3:0]
isi_cpr_reset_n
gpio_isi_din[3:0]
cpu_adr[n:2]
cpu_dataout[31:0]
scb_cpu_data[31:0]
cpu_scb_sel
cpu_rwn
cpu_acode[1:0]
scb_cpu_rdy
scb_cpu_berr
scb_cpu_debug_valid
dma_icu_irq
isi_icu_irq
usb_icu_irq[1:0]
scb_diu_wadr[21:5]
scb_diu_data[63:0]
scb_diu_wreq
diu_scb_wack
scb_diu_wvalid
scb_diu_wmask[7:0]
scb_diu_rreq
scb_diu_radr[21:5]
diu_scb_rack
diu_scb_rvalid
diu_data[63:0]
prst_n
pclk

FIG. 28

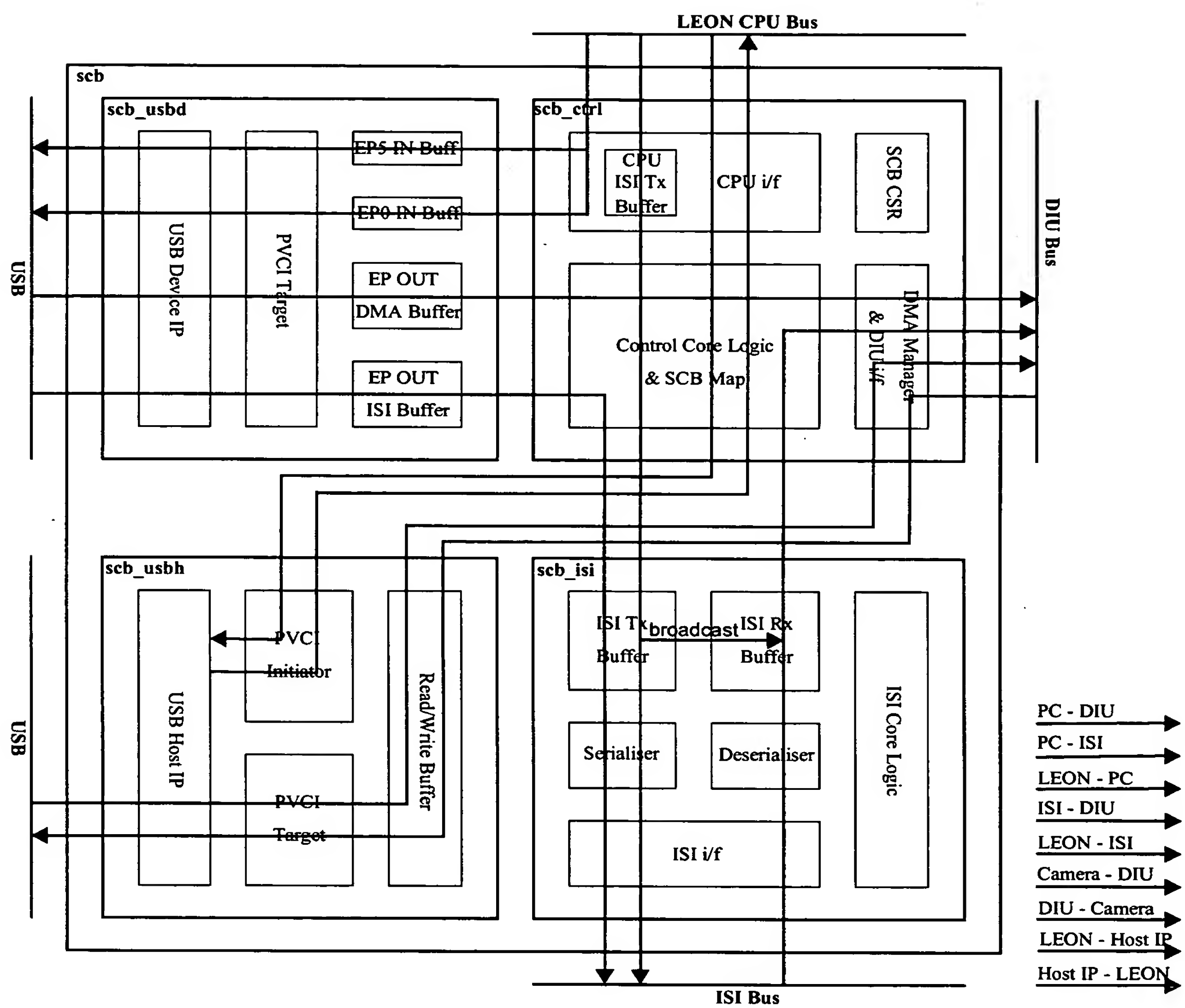

LEON CPU Bus
scb
scb_usbd
USB Device IP
PVCI Target
EP5 IN Buff
EP0 IN Buff
EP OUT
DMA Buffer
EP OUT
ISI Buffer
scb_ctrl
CPU
ISI Tx
Buffer
CPU i/f
SCB CSR
Control Core Logic
& SCB Map
DMA Manager
& DIU i/f
DIU Bus
USB
scb_usbh
USB Host IP
PVCI
Initiator
PVCI
Target
Read/Write Buffer
USB
scb_isi
ISI Tx
Buffer
broadcast
ISI Rx
Buffer
Serialiser
Deserialiser
ISI i/f
ISI Core Logic
ISI Bus
PC - DIU
PC - ISI
LEON - PC
ISI - DIU
LEON - ISI
Camera - DIU
DIU - Camera
LEON - Host IP
Host IP - LEON

FIG. 29

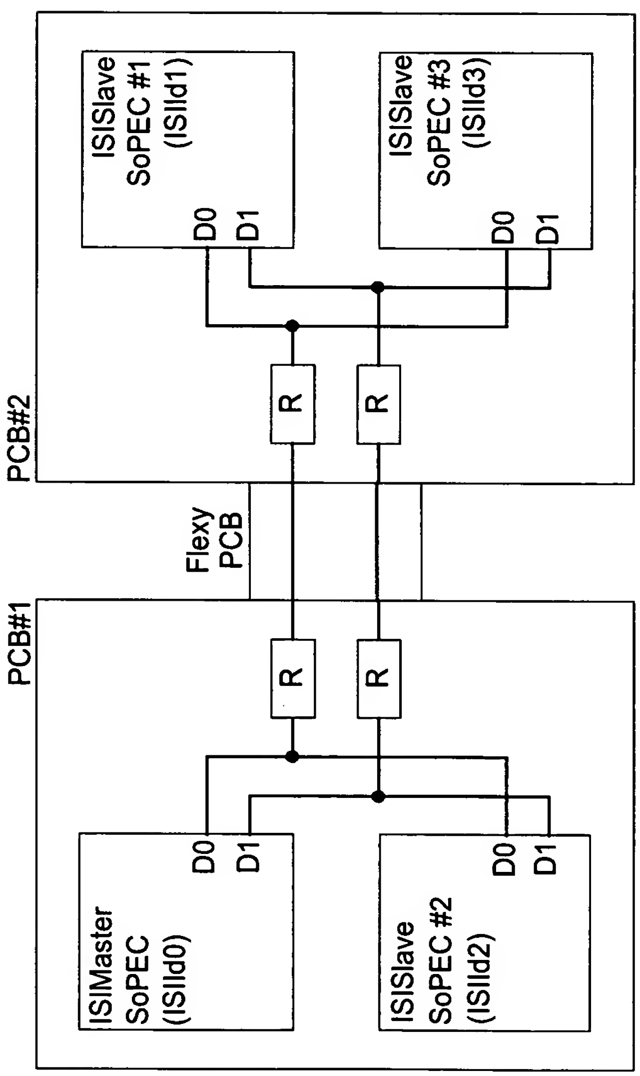

PCB#1
PCB#2
Flexy PCB
ISIMaster SoPEC (ISIId0)
ISISlave SoPEC #2 (ISIId2)
ISISlave SoPEC #1 (ISIId1)
ISISlave SoPEC #3 (ISIId3)
D0
D1
R

FIG. 30

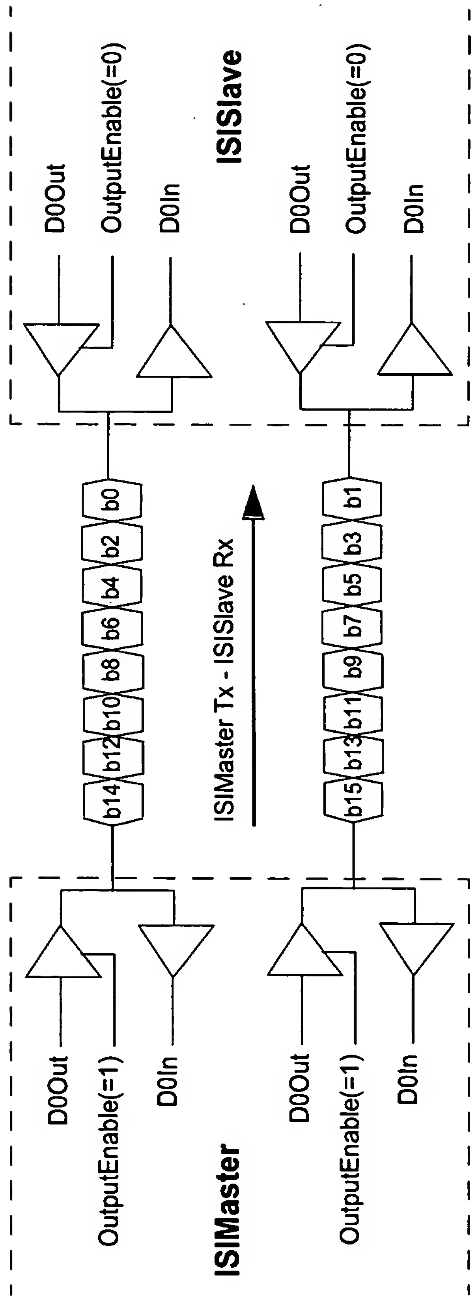
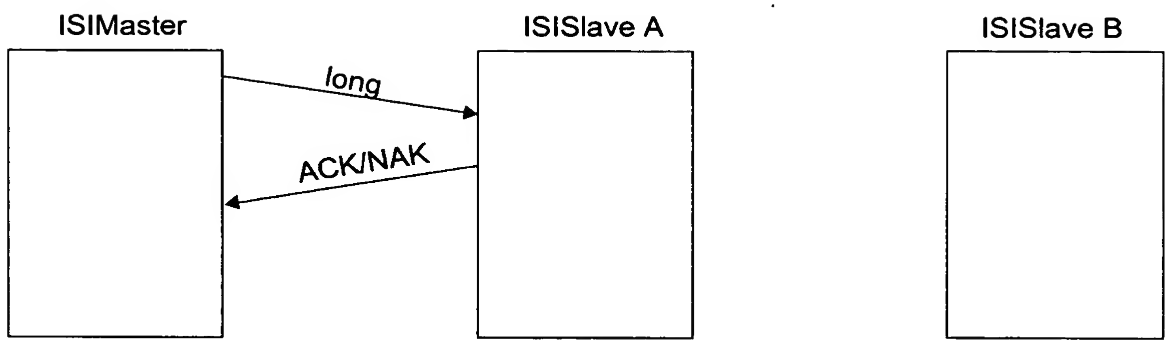
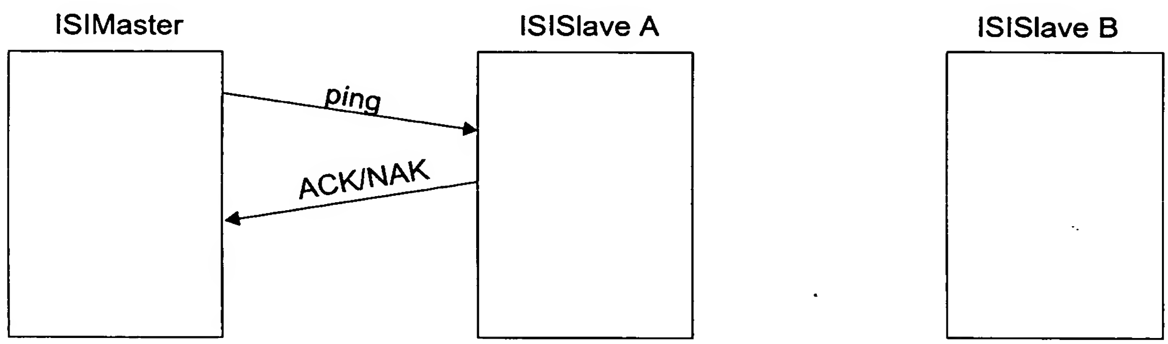

ISISlave
D0Out
OutputEnable(=0)
D0In
D0Out
OutputEnable(=0)
D0In
b0
b2
b4
b6
b8
b10
b12
b14
b1
b3
b5
b7
b9
b11
b13
b15
ISIMaster Tx - ISISlave Rx
ISIMaster
D0Out
OutputEnable(=1)
D0In
D0Out
OutputEnable(=1)
D0In

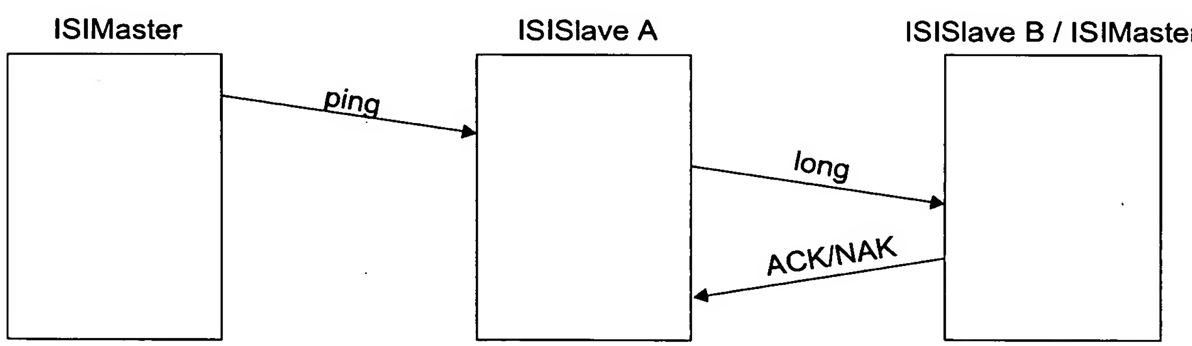
FIG. 31



Transaction 1: Long packet to an addressed ISISlave



Transaction 2: Ping packet to an addressed ISISlave. ISISlave has nothing to send



Transaction 3: Ping packet to an addressed ISISlave. ISISlaveA responds with a long packet to ISISlaveB (or the ISIMaster) and ISISlaveB (or the ISIMaster) responds with an ACK or NAK.

FIG. 32

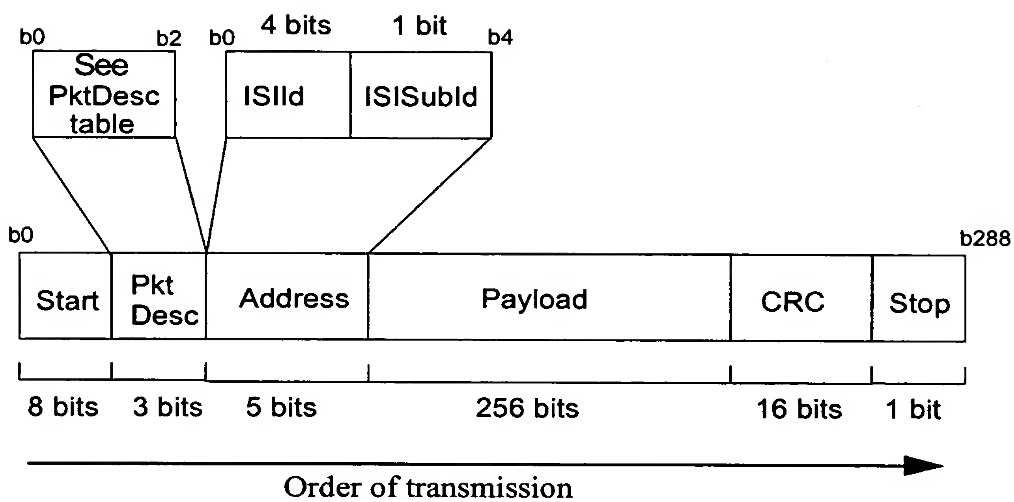

b0
b2
b0
4 bits
1 bit
b4
See PktDesc table
ISIId
ISISubId
b0
b288
Start
Pkt Desc
Address
Payload
CRC
Stop
8 bits
3 bits
5 bits
256 bits
16 bits
1 bit
Order of transmission

FIG. 33

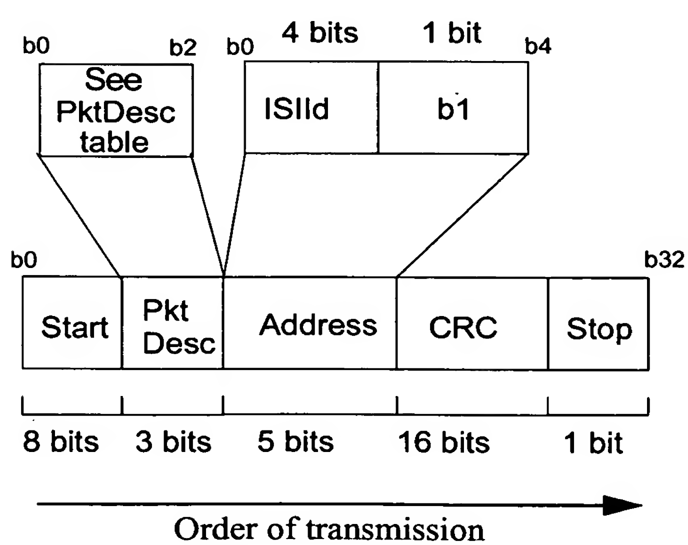

b0
b2
b0
4 bits
1 bit
b4
See PktDesc table
ISIId
b1
b0
b32
Start
Pkt Desc
Address
CRC
Stop
8 bits
3 bits
5 bits
16 bits
1 bit
Order of transmission

FIG. 34

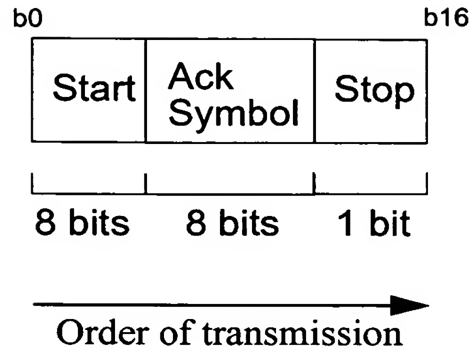

b0
b16
Start
Ack Symbol
Stop
8 bits
8 bits
1 bit
Order of transmission

FIG. 35

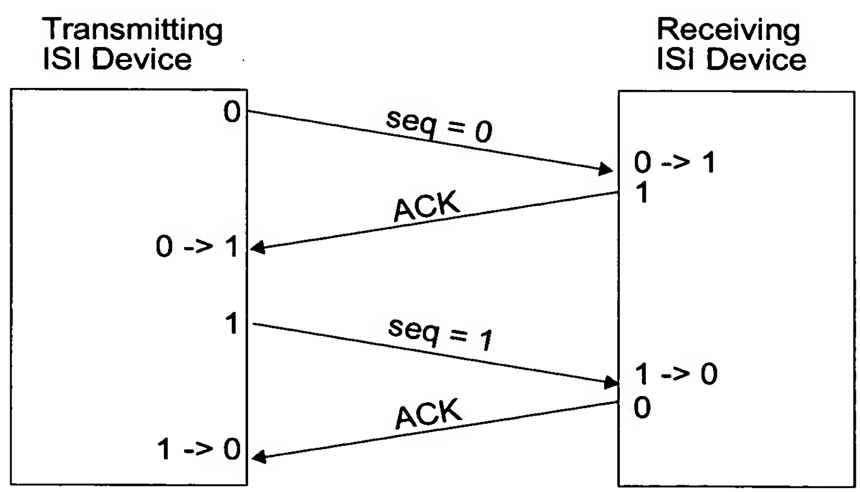

Transmitting ISI Device
Receiving ISI Device
0
seq = 0
0 -> 1
1
ACK
0 -> 1
1
seq = 1
1 -> 0
0
ACK
1 -> 0

FIG. 36

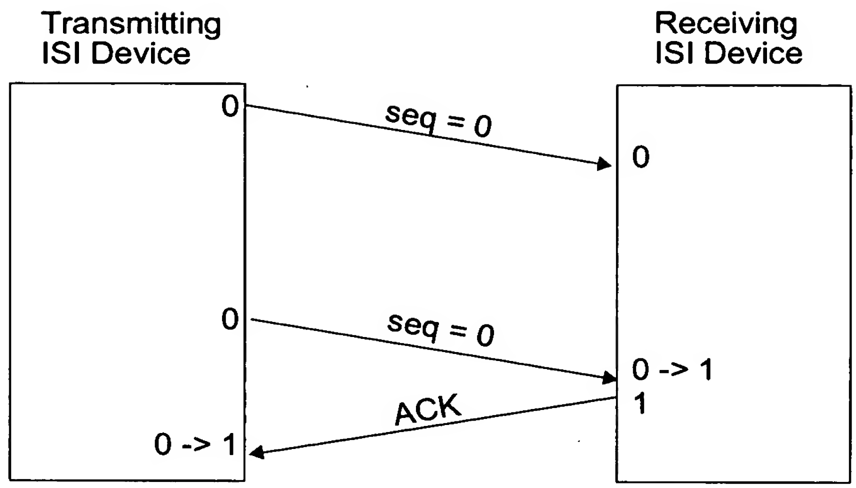

Transmitting
ISI Device
Receiving
ISI Device
0
seq = 0
0
0
seq = 0
0 -> 1
ACK
1
0 -> 1

FIG. 37

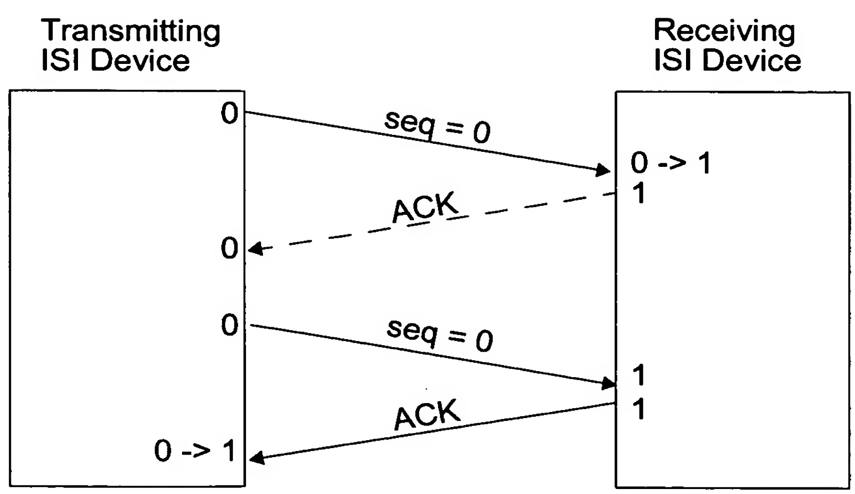

Transmitting
ISI Device
Receiving
ISI Device
0
seq = 0
0 -> 1
ACK
1
0
0
seq = 0
1
ACK
1
0 -> 1

FIG. 38

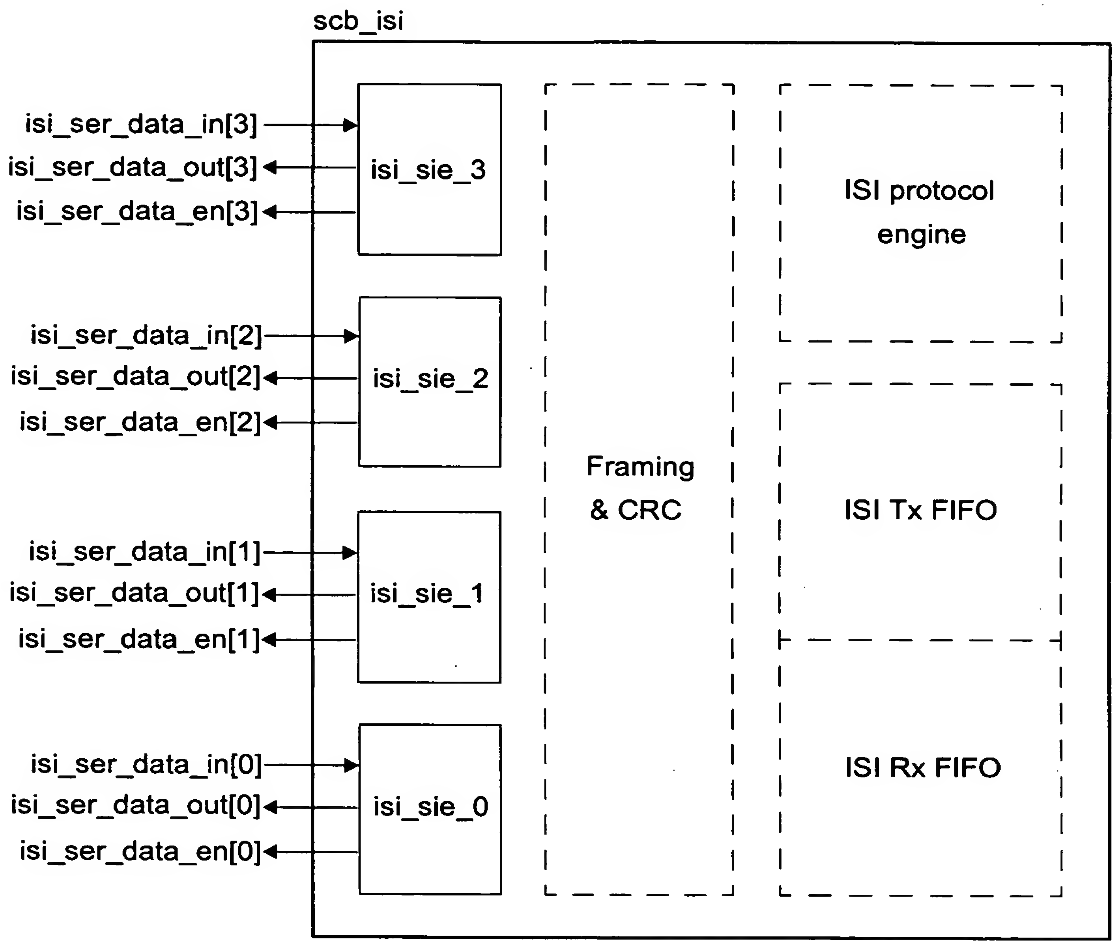

scb_isi
isi_ser_data_in[3]
isi_ser_data_out[3]
isi_ser_data_en[3]
isi_sie_3
isi_ser_data_in[2]
isi_ser_data_out[2]
isi_ser_data_en[2]
isi_sie_2
isi_ser_data_in[1]
isi_ser_data_out[1]
isi_ser_data_en[1]
isi_sie_1
isi_ser_data_in[0]
isi_ser_data_out[0]
isi_ser_data_en[0]
isi_sie_0
Framing & CRC
ISI protocol engine
ISI Tx FIFO
ISI Rx FIFO

FIG. 39

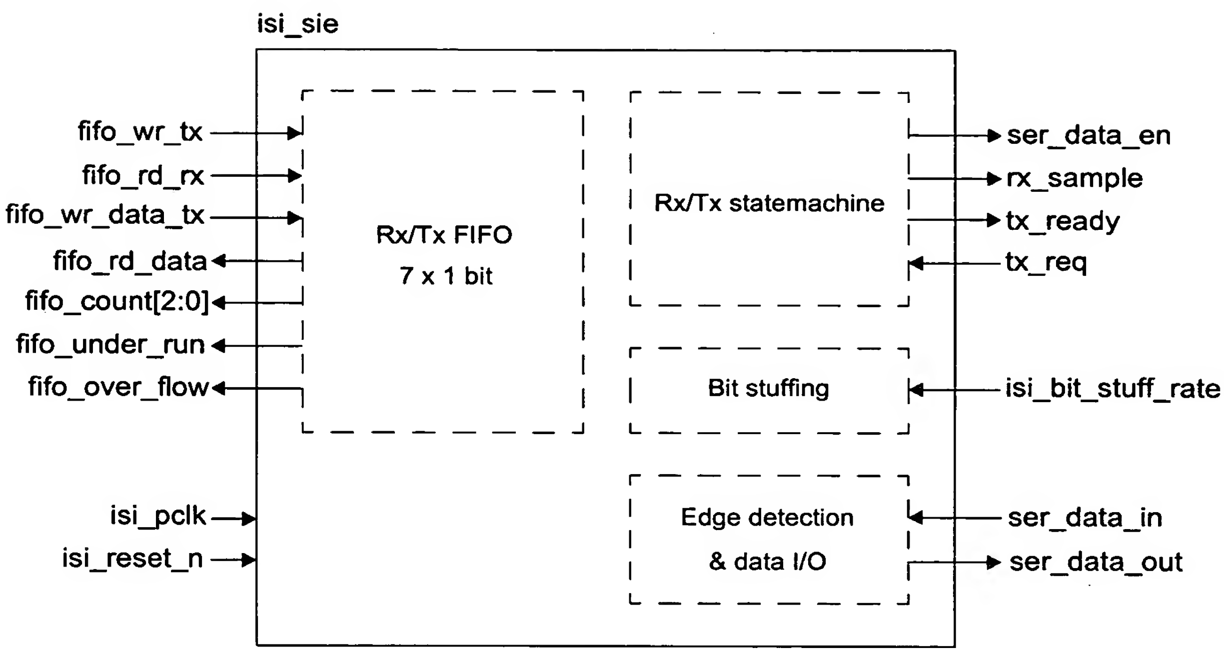

isi_sie
fifo_wr_tx
fifo_rd_rx
fifo_wr_data_tx
fifo_rd_data
fifo_count[2:0]
fifo_under_run
fifo_over_flow
Rx/Tx FIFO
7 x 1 bit
Rx/Tx statemachine
ser_data_en
rx_sample
tx_ready
tx_req
Bit stuffing
isi_bit_stuff_rate
isi_pclk
isi_reset_n
Edge detection
& data I/O
ser_data_in
ser_data_out

FIG. 40

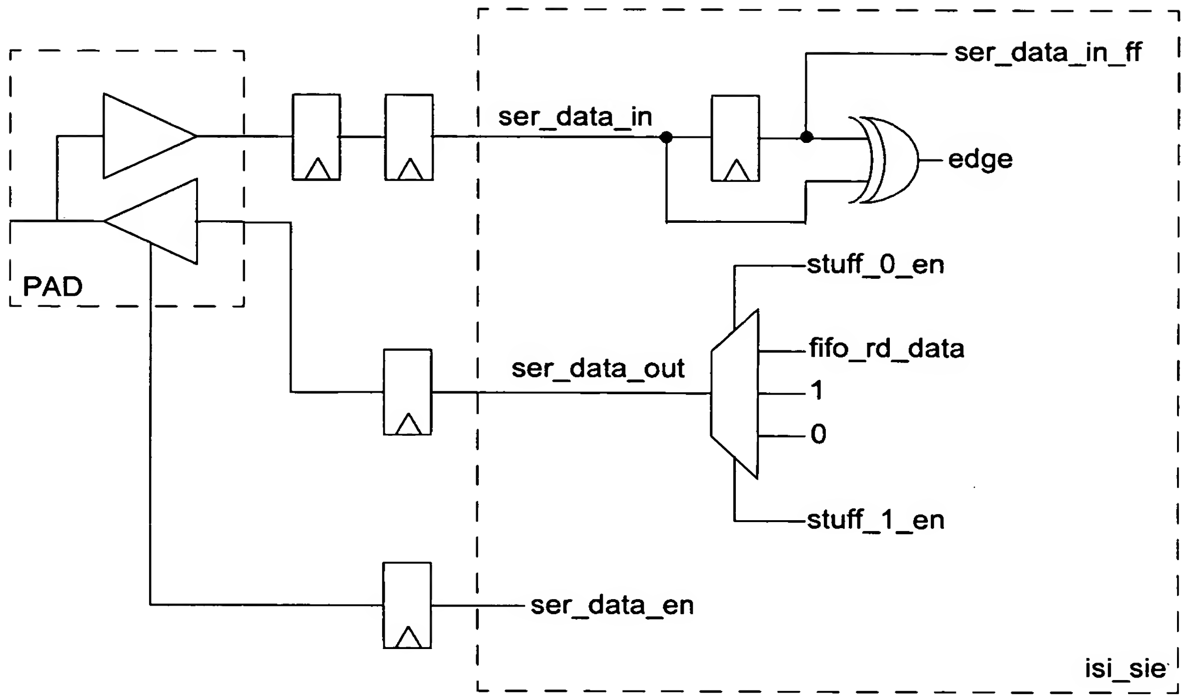

ser_data_in_ff
ser_data_in
edge
PAD
stuff_0_en
fifo_rd_data
ser_data_out
1
0
stuff_1_en
ser_data_en
isi_sie

FIG. 41

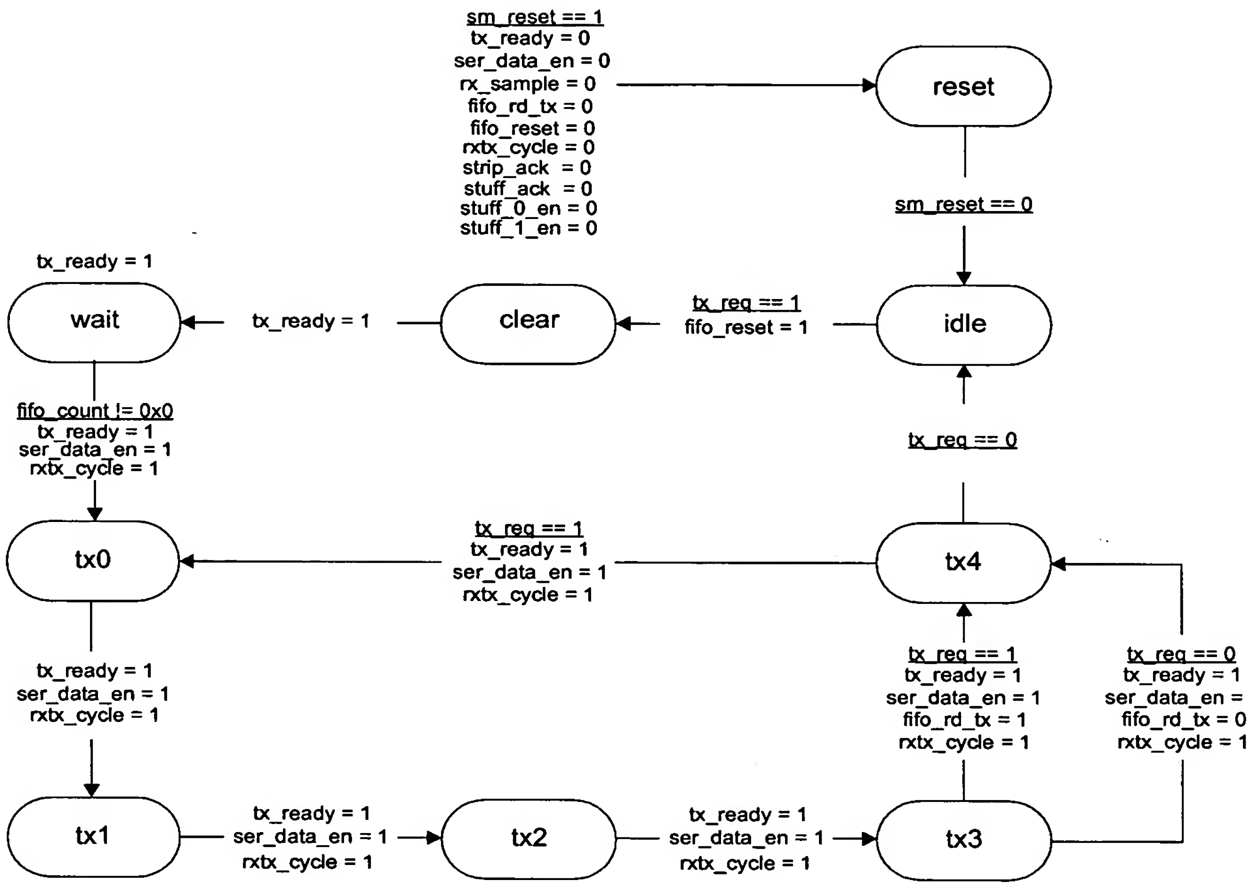

sm_reset == 1
tx_ready = 0
ser_data_en = 0
rx_sample = 0
fifo_rd_tx = 0
fifo_reset = 0
rxtx_cycle = 0
strip_ack = 0
stuff_ack = 0
stuff_0_en = 0
stuff_1_en = 0
reset
sm_reset == 0
tx_ready = 1
wait
tx_ready = 1
clear
tx_req == 1
fifo_reset = 1
idle
fifo_count != 0x0
tx_ready = 1
ser_data_en = 1
rxtx_cycle = 1
tx_req == 0
tx0
tx_req == 1
tx_ready = 1
ser_data_en = 1
rxtx_cycle = 1
tx4
tx_ready = 1
ser_data_en = 1
rxtx_cycle = 1
tx_req == 1
tx_ready = 1
ser_data_en = 1
fifo_rd_tx = 1
rxtx_cycle = 1
tx_req == 0
tx_ready = 1
ser_data_en = 1
fifo_rd_tx = 0
rxtx_cycle = 1
tx1
tx_ready = 1
ser_data_en = 1
rxtx_cycle = 1
tx2
tx_ready = 1
ser_data_en = 1
rxtx_cycle = 1
tx3

FIG. 42

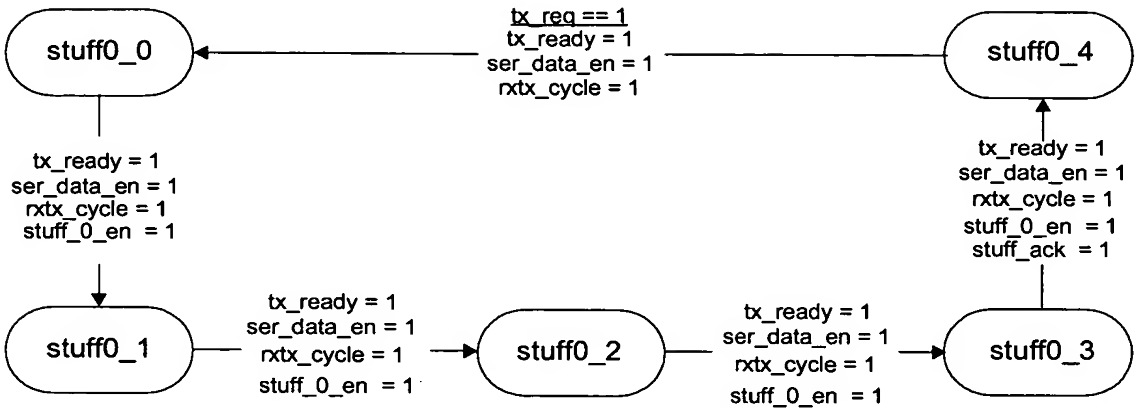

stuff0_0
tx_req == 1
tx_ready = 1
ser_data_en = 1
rxtx_cycle = 1
stuff0_4
tx_ready = 1
ser_data_en = 1
rxtx_cycle = 1
stuff_0_en = 1
tx_ready = 1
ser_data_en = 1
rxtx_cycle = 1
stuff_0_en = 1
stuff_ack = 1
stuff0_1
tx_ready = 1
ser_data_en = 1
rxtx_cycle = 1
stuff_0_en = 1
stuff0_2
tx_ready = 1
ser_data_en = 1
rxtx_cycle = 1
stuff_0_en = 1
stuff0_3

FIG. 43

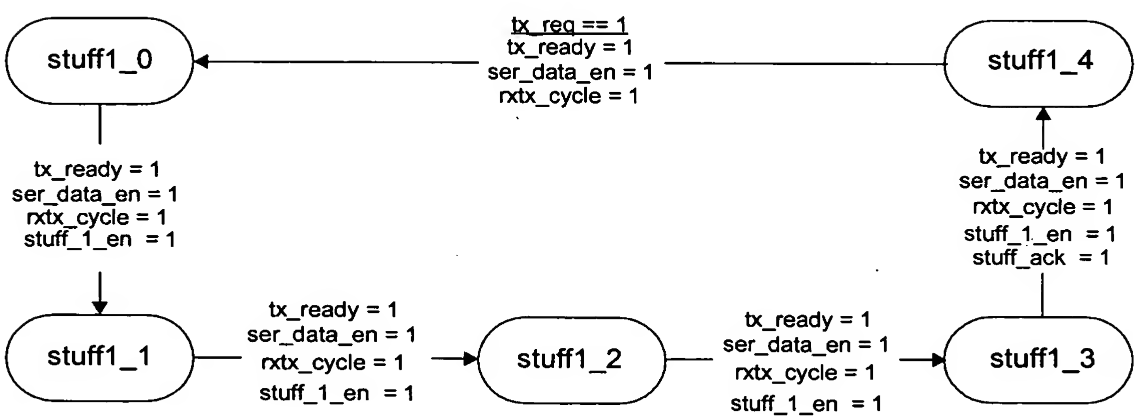

stuff1_0
tx_req == 1
tx_ready = 1
ser_data_en = 1
rxtx_cycle = 1
stuff1_4
tx_ready = 1
ser_data_en = 1
rxtx_cycle = 1
stuff_1_en = 1
tx_ready = 1
ser_data_en = 1
rxtx_cycle = 1
stuff_1_en = 1
stuff_ack = 1
stuff1_1
tx_ready = 1
ser_data_en = 1
rxtx_cycle = 1
stuff_1_en = 1
stuff1_2
tx_ready = 1
ser_data_en = 1
rxtx_cycle = 1
stuff_1_en = 1
stuff1_3

FIG. 44

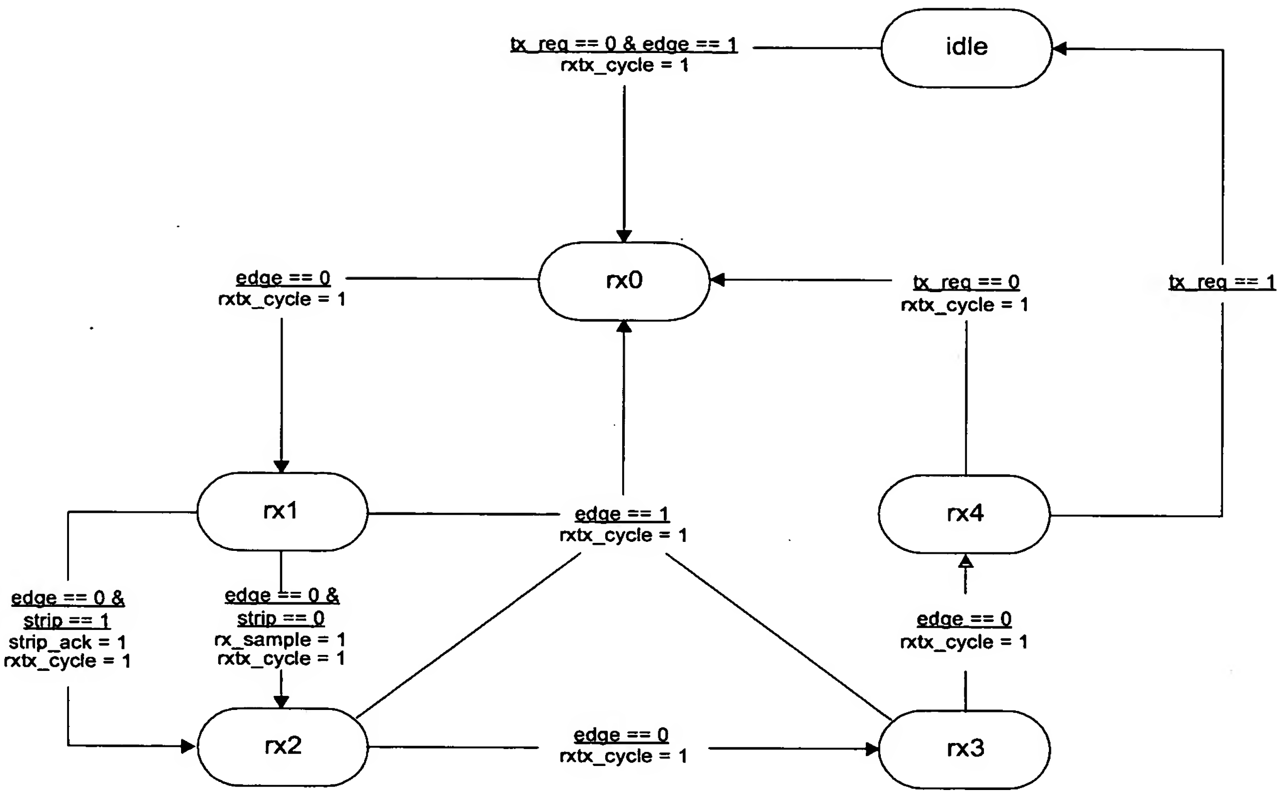

idle
tx_req == 0 & edge == 1
rxtx_cycle = 1
rx0
edge == 0
rxtx_cycle = 1
tx_req == 0
rxtx_cycle = 1
tx_req == 1
rx1
edge == 1
rxtx_cycle = 1
rx4
edge == 0 &
strip == 1
strip_ack = 1
rxtx_cycle = 1
edge == 0 &
strip == 0
rx_sample = 1
rxtx_cycle = 1
edge == 0
rxtx_cycle = 1
rx2
edge == 0
rxtx_cycle = 1
rx3

FIG. 45

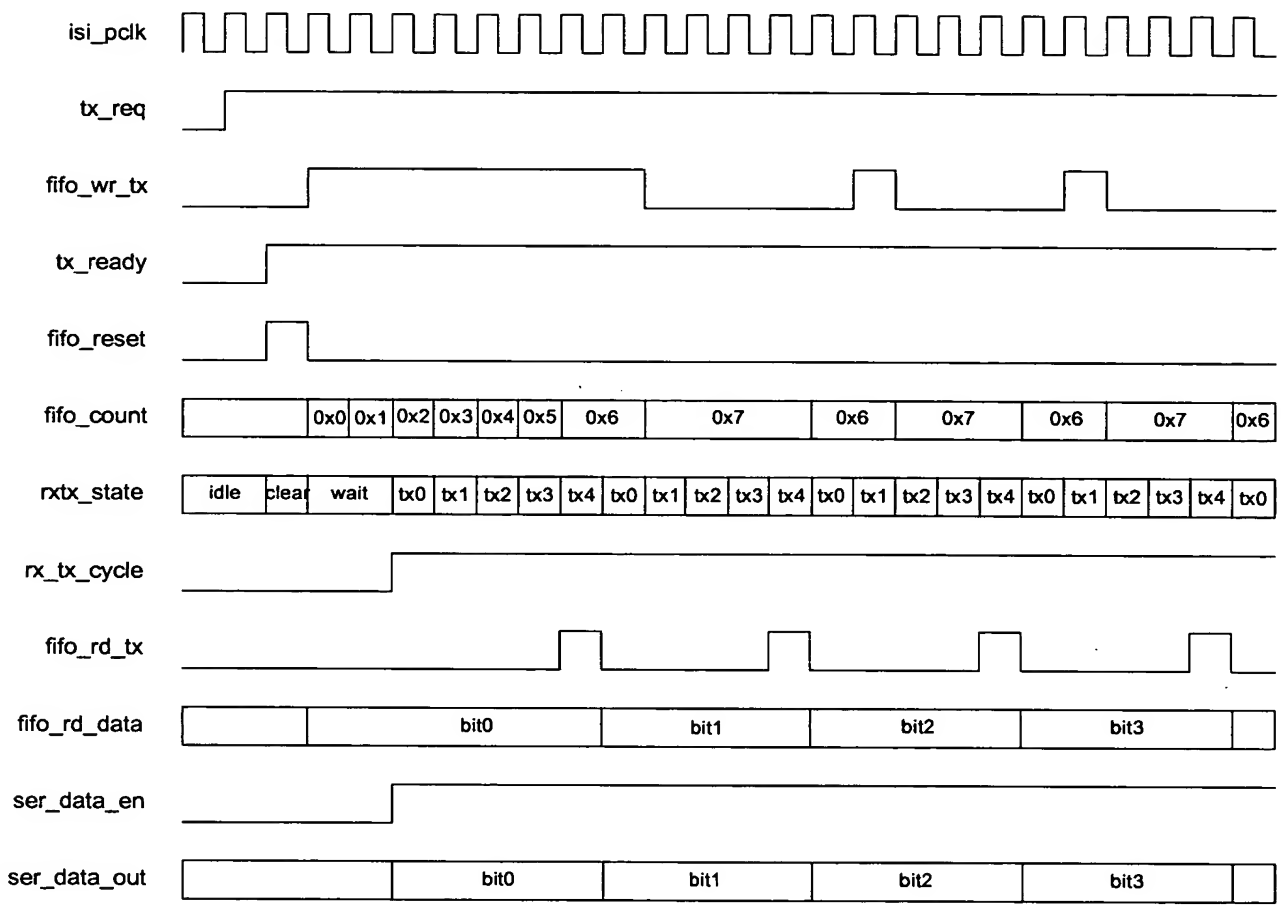

isi_pclk
tx_req
fifo_wr_tx
tx_ready
fifo_reset
fifo_count
0x0 0x1 0x2 0x3 0x4 0x5 0x6 0x7 0x6 0x7 0x6 0x7 0x6
rxtx_state
idle clear wait tx0 tx1 tx2 tx3 tx4 tx0 tx1 tx2 tx3 tx4 tx0 tx1 tx2 tx3 tx4 tx0 tx1 tx2 tx3 tx4 tx0
rx_tx_cycle
fifo_rd_tx
fifo_rd_data
bit0 bit1 bit2 bit3
ser_data_en
ser_data_out
bit0 bit1 bit2 bit3

FIG. 46

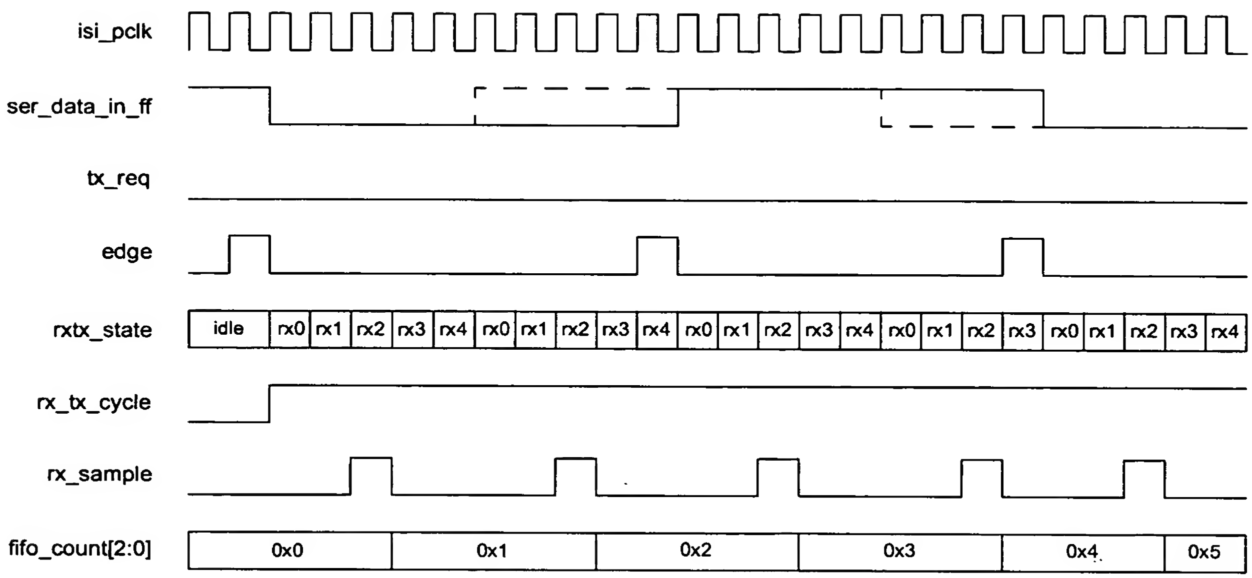

isi_pclk
ser_data_in_ff
tx_req
edge
rxtx_state
idle
rx0
rx1
rx2
rx3
rx4
rx_tx_cycle
rx_sample
fifo_count[2:0]
0x0
0x1
0x2
0x3
0x4
0x5

FIG. 47

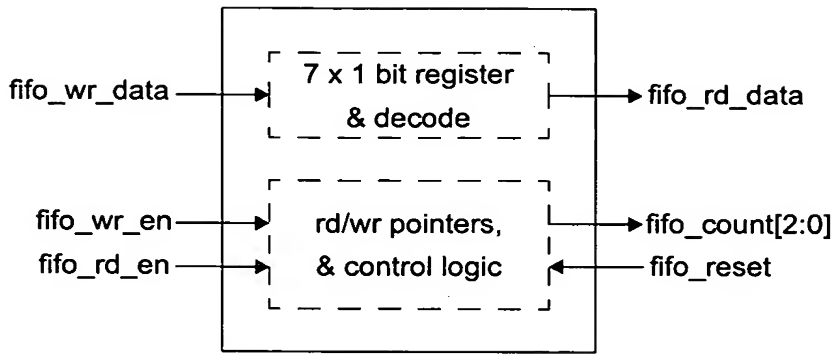

fifo_wr_data
7 x 1 bit register
& decode
fifo_rd_data
fifo_wr_en
fifo_rd_en
rd/wr pointers,
& control logic
fifo_count[2:0]
fifo_reset

FIG. 48

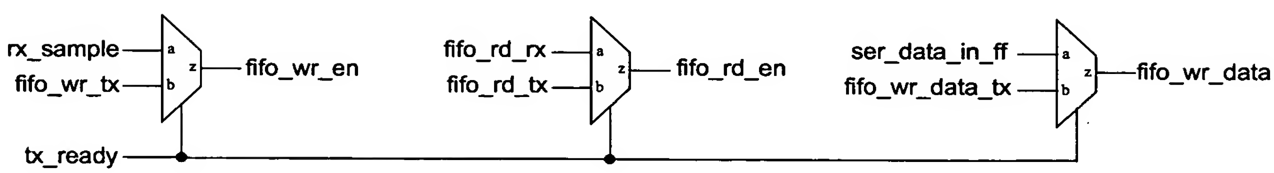

rx_sample
fifo_wr_tx
fifo_wr_en
fifo_rd_rx
fifo_rd_tx
fifo_rd_en
ser_data_in_ff
fifo_wr_data_tx
fifo_wr_data
tx_ready

FIG. 49

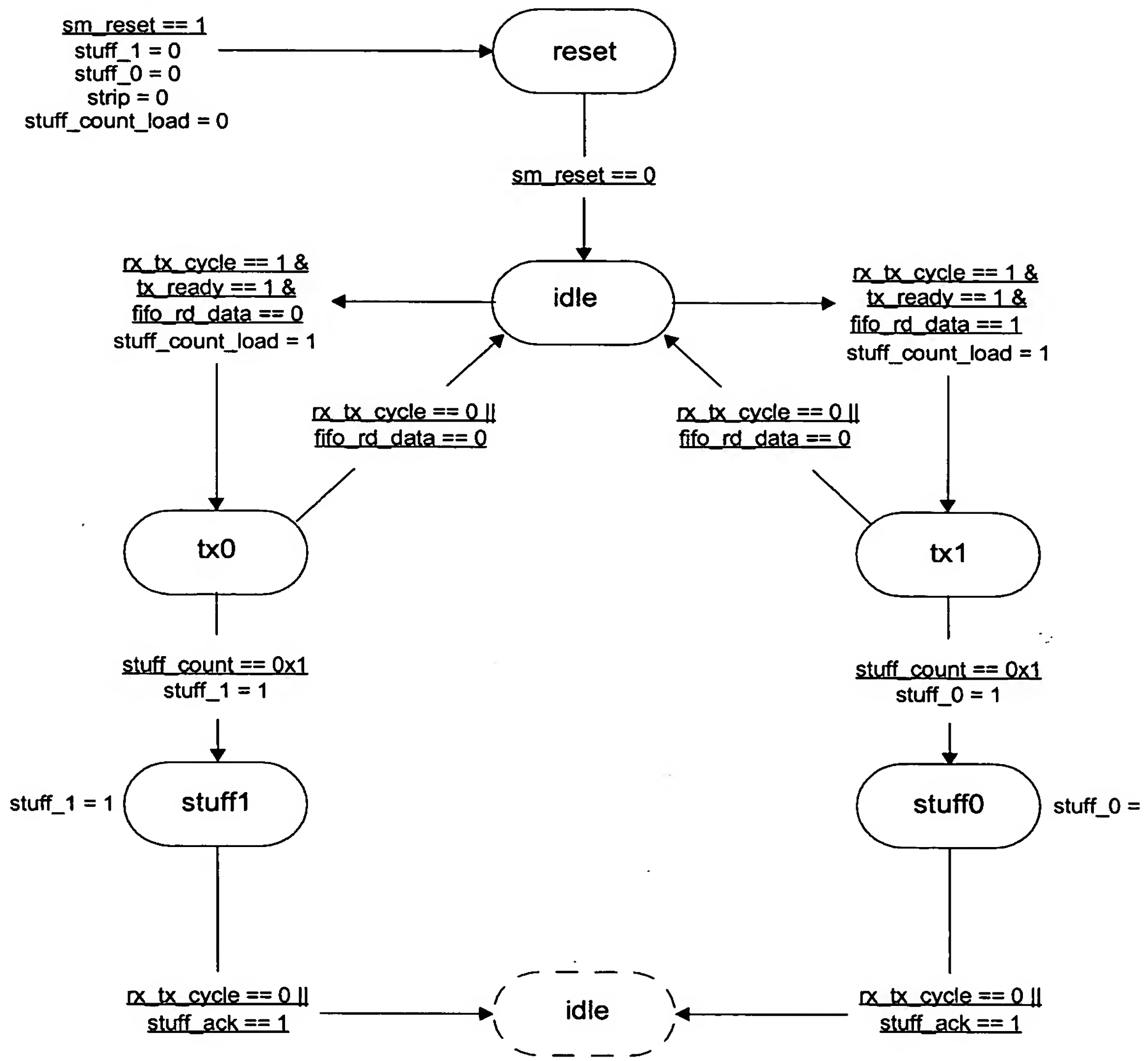

sm_reset == 1
stuff_1 = 0
stuff_0 = 0
strip = 0
stuff_count_load = 0
reset
sm_reset == 0
idle
rx_tx_cycle == 1 &
tx_ready == 1 &
fifo_rd_data == 0
stuff_count_load = 1
rx_tx_cycle == 1 &
tx_ready == 1 &
fifo_rd_data == 1
stuff_count_load = 1
rx_tx_cycle == 0 ||
fifo_rd_data == 0
rx_tx_cycle == 0 ||
fifo_rd_data == 0
tx0
tx1
stuff_count == 0x1
stuff_1 = 1
stuff_count == 0x1
stuff_0 = 1
stuff_1 = 1
stuff1
stuff0
stuff_0 = 1
rx_tx_cycle == 0 ||
stuff_ack == 1
idle
rx_tx_cycle == 0 ||
stuff_ack == 1

FIG. 50

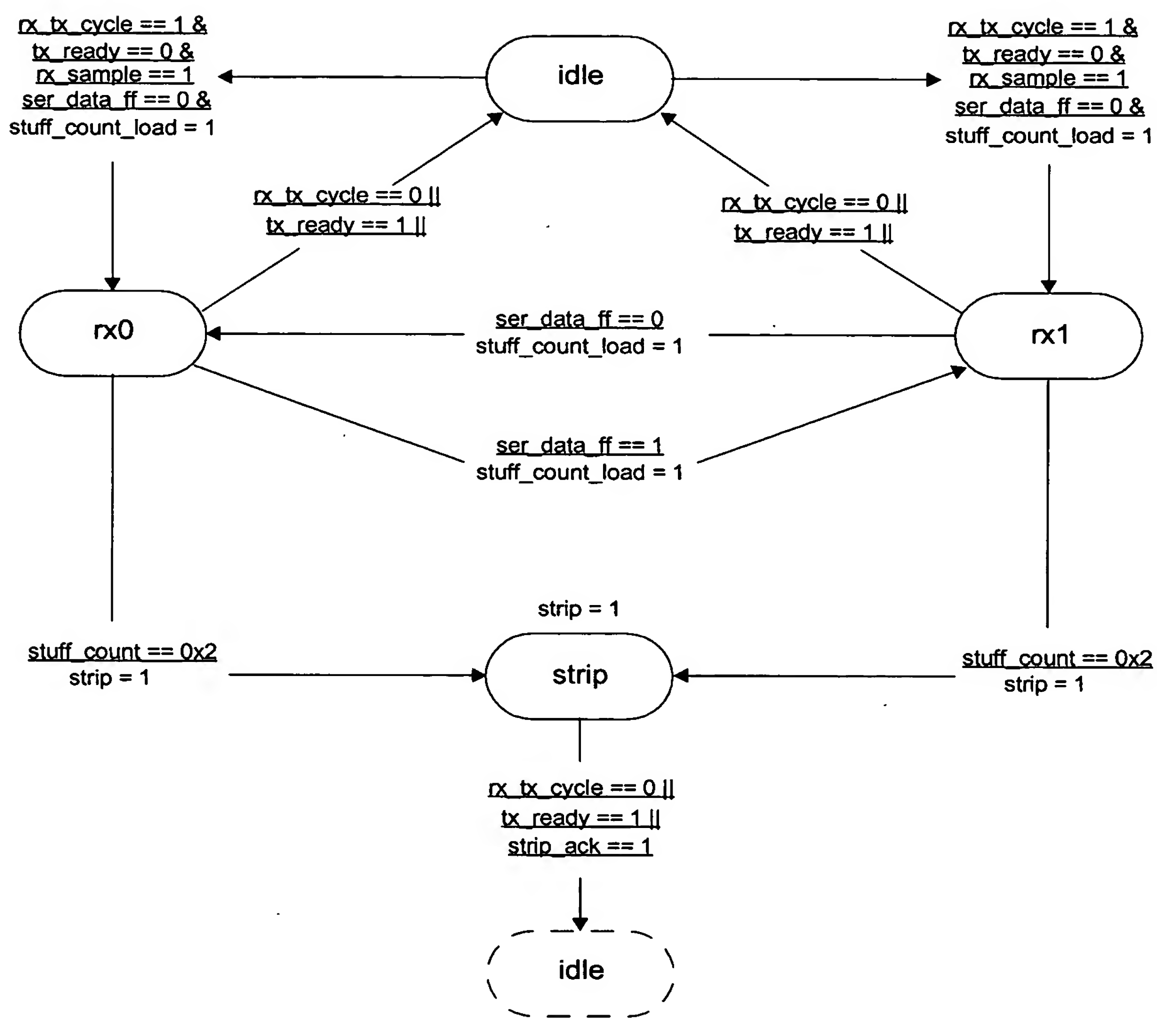

idle
rx_tx_cycle == 1 &
tx_ready == 0 &
rx_sample == 1
ser_data_ff == 0 &
stuff_count_load = 1
rx_tx_cycle == 1 &
tx_ready == 0 &
rx_sample == 1
ser_data_ff == 0 &
stuff_count_load = 1
rx_tx_cycle == 0 ||
tx_ready == 1 ||
rx_tx_cycle == 0 ||
tx_ready == 1 ||
rx0
ser_data_ff == 0
stuff_count_load = 1
rx1
ser_data_ff == 1
stuff_count_load = 1
strip = 1
stuff_count == 0x2
strip = 1
strip
stuff_count == 0x2
strip = 1
rx_tx_cycle == 0 ||
tx_ready == 1 ||
strip_ack == 1
idle

FIG. 51

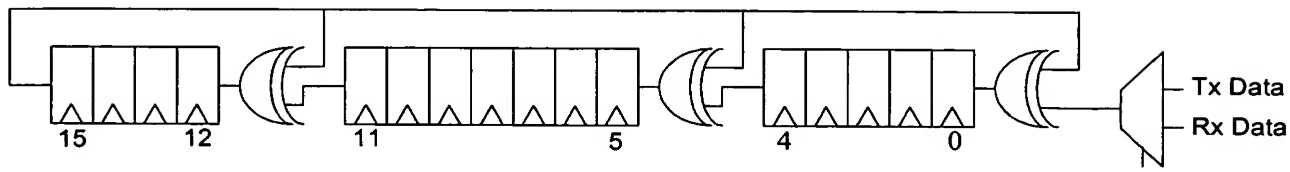

15
12
11
5
4
0
Tx Data
Rx Data

FIG. 52

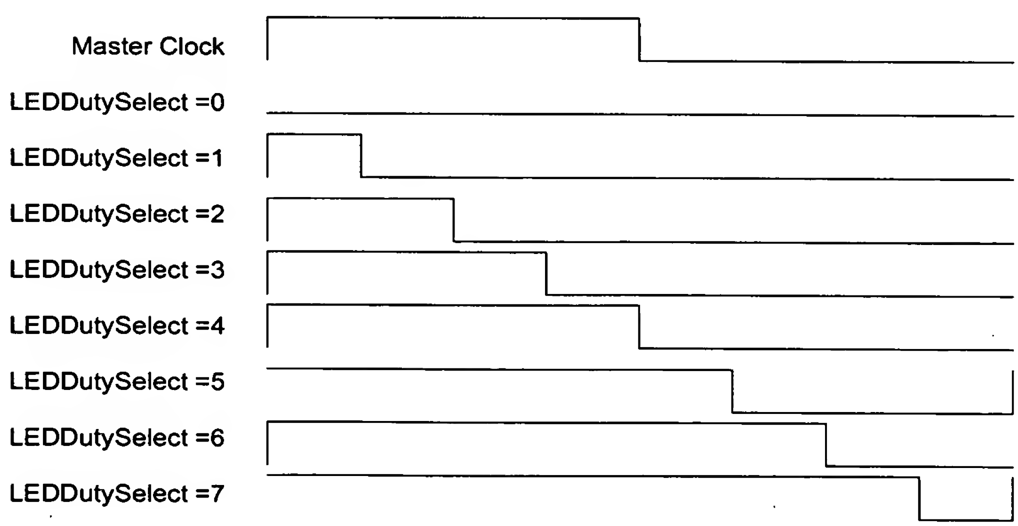

Master Clock
LEDDutySelect =0
LEDDutySelect =1
LEDDutySelect =2
LEDDutySelect =3
LEDDutySelect =4
LEDDutySelect =5
LEDDutySelect =6
LEDDutySelect =7

FIG. 54

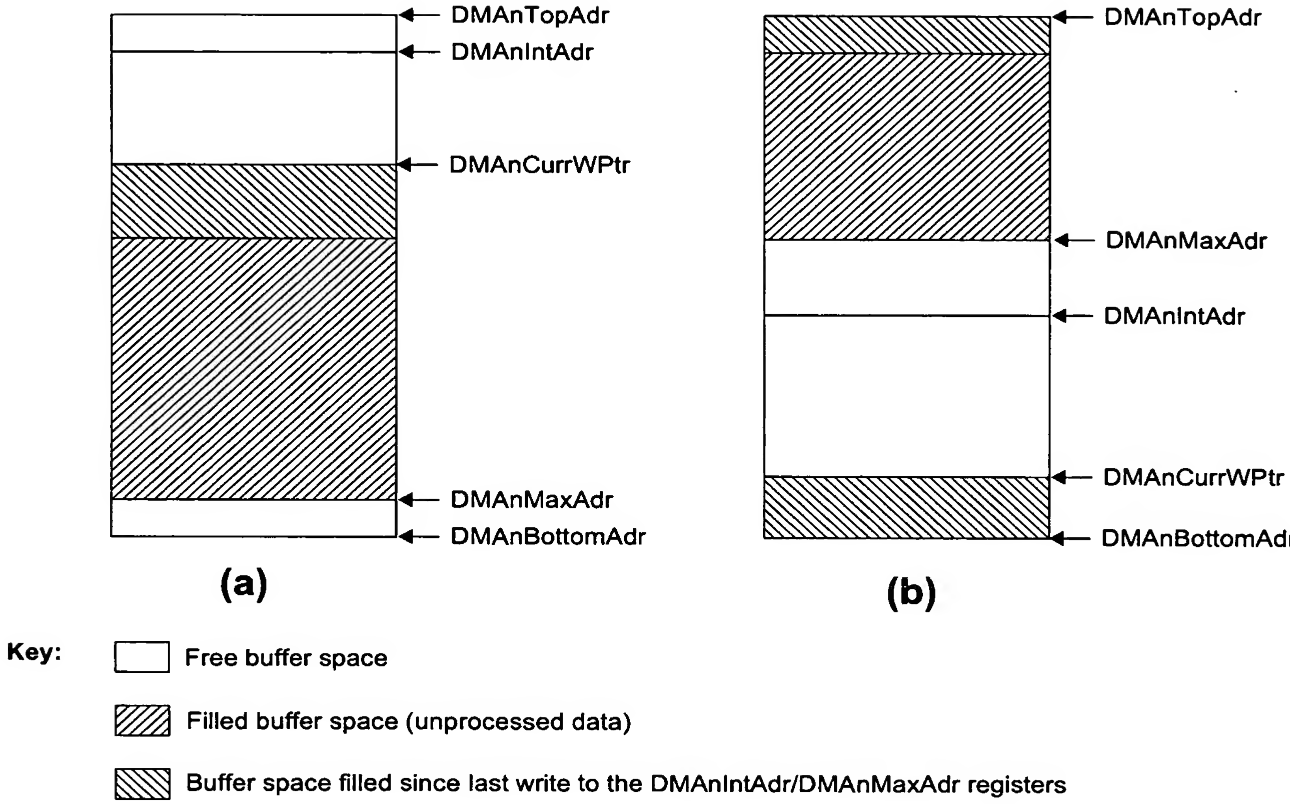

DMAnTopAdr
DMAnIntAdr
DMAnCurrWPtr
DMAnMaxAdr
DMAnBottomAdr
(a)
DMAnTopAdr
DMAnMaxAdr
DMAnIntAdr
DMAnCurrWPtr
DMAnBottomAdr
(b)
Key:
Free buffer space
Filled buffer space (unprocessed data)
Buffer space filled since last write to the DMAnIntAdr/DMAnMaxAdr registers

FIG. 53

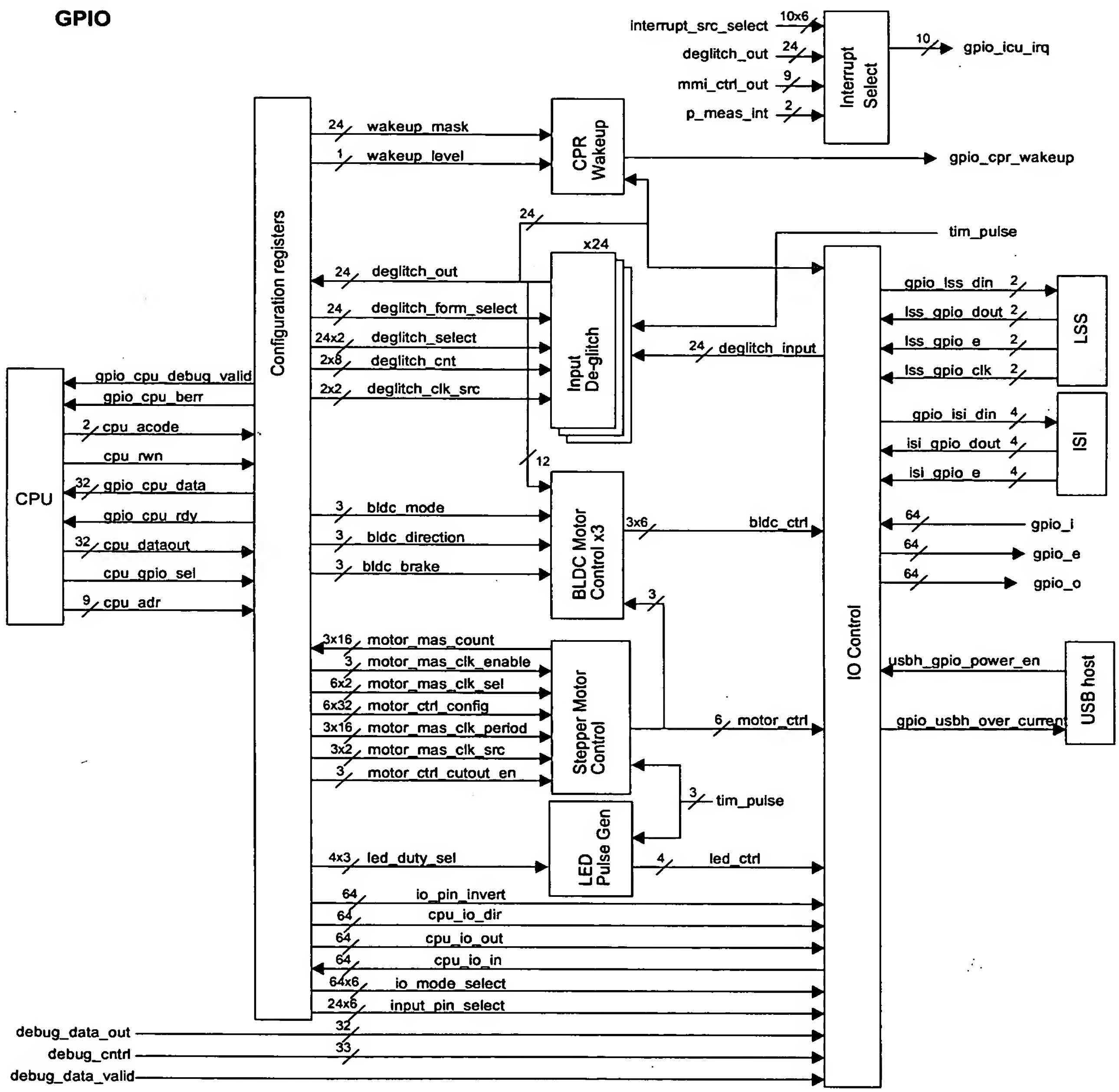
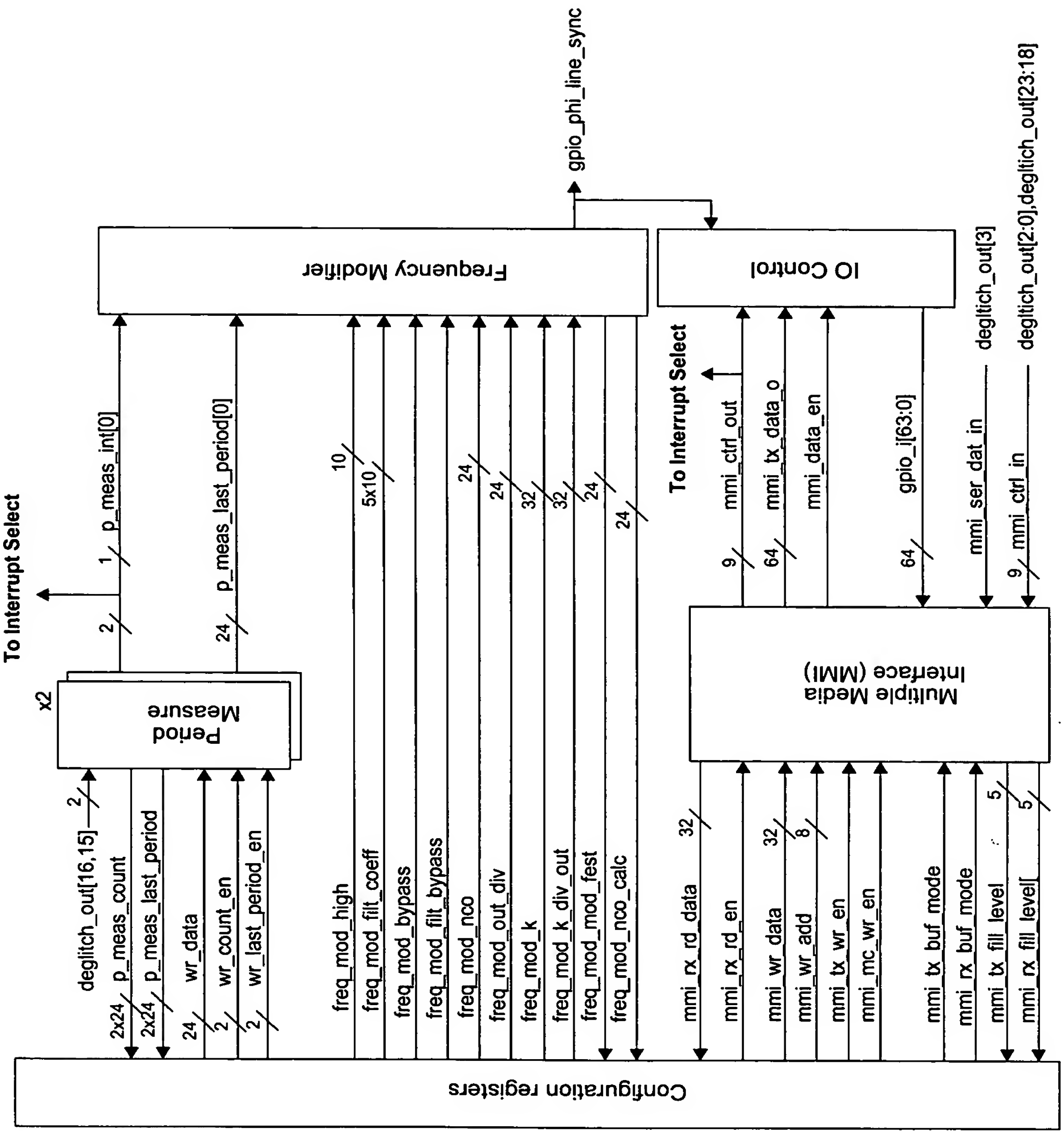

GPIO
interrupt_src_select
10x6
deglitch_out
24
mmi_ctrl_out
9
p_meas_int
2
Interrupt Select
10
gpio_icu_irq
Configuration registers
24
wakeup_mask
1
wakeup_level
CPR Wakeup
gpio_cpr_wakeup
24
tim_pulse
x24
24
deglitch_out
24
deglitch_form_select
24x2
deglitch_select
2x8
deglitch_cnt
2x2
deglitch_clk_src
Input De-glitch
24
deglitch_input
IO Control
gpio_lss_din
lss_gpio_dout
lss_gpio_e
lss_gpio_clk
LSS
gpio_isi_din
isi_gpio_dout
isi_gpio_e
ISI
CPU
gpio_cpu_debug_valid
gpio_cpu_berr
cpu_acode
cpu_rwn
32
gpio_cpu_data
gpio_cpu_rdy
cpu_dataout
cpu_gpio_sel
9
cpu_adr
12
3
bldc_mode
bldc_direction
bldc_brake
BLDC Motor Control x3
3x6
bldc_ctrl
64
gpio_i
gpio_e
gpio_o
3x16
motor_mas_count
motor_mas_clk_enable
6x2
motor_mas_clk_sel
6x32
motor_ctrl_config
motor_mas_clk_period
3x2
motor_mas_clk_src
motor_ctrl_cutout_en
Stepper Motor Control
6
motor_ctrl
usbh_gpio_power_en
gpio_usbh_over_current
USB host
tim_pulse
4x3
led_duty_sel
LED Pulse Gen
4
led_ctrl
io_pin_invert
cpu_io_dir
cpu_io_out
cpu_io_in
64x6
io_mode_select
24x6
input_pin_select
debug_data_out
debug_cntrl
debug_data_valid
33

FIG. 55



Configuration registers
Period Measure
x2
To Interrupt Select
Frequency Modifier
IO Control
Multiple Media Interface (MMI)
gpio_phi_line_sync
deglitch_out[16,15]
p_meas_count
p_meas_last_period
wr_data
wr_count_en
wr_last_period_en
p_meas_int[0]
p_meas_last_period[0]
freq_mod_high
freq_mod_filt_coeff
freq_mod_bypass
freq_mod_filt_bypass
freq_mod_nco
freq_mod_out_div
freq_mod_k
freq_mod_k_div_out
freq_mod_mod_fest
freq_mod_nco_calc
mmi_rx_rd_data
mmi_rx_rd_en
mmi_wr_data
mmi_wr_add
mmi_tx_wr_en
mmi_mc_wr_en
mmi_tx_buf_mode
mmi_rx_buf_mode
mmi_tx_fill_level
mmi_rx_fill_level
mmi_ctrl_out
mmi_tx_data_o
mmi_data_en
gpio_i[63:0]
mmi_ser_dat_in
mmi_ctrl_in
deglitch_out[3]
deglitch_out[2:0],deglitch_out[23:18]

FIG. 56

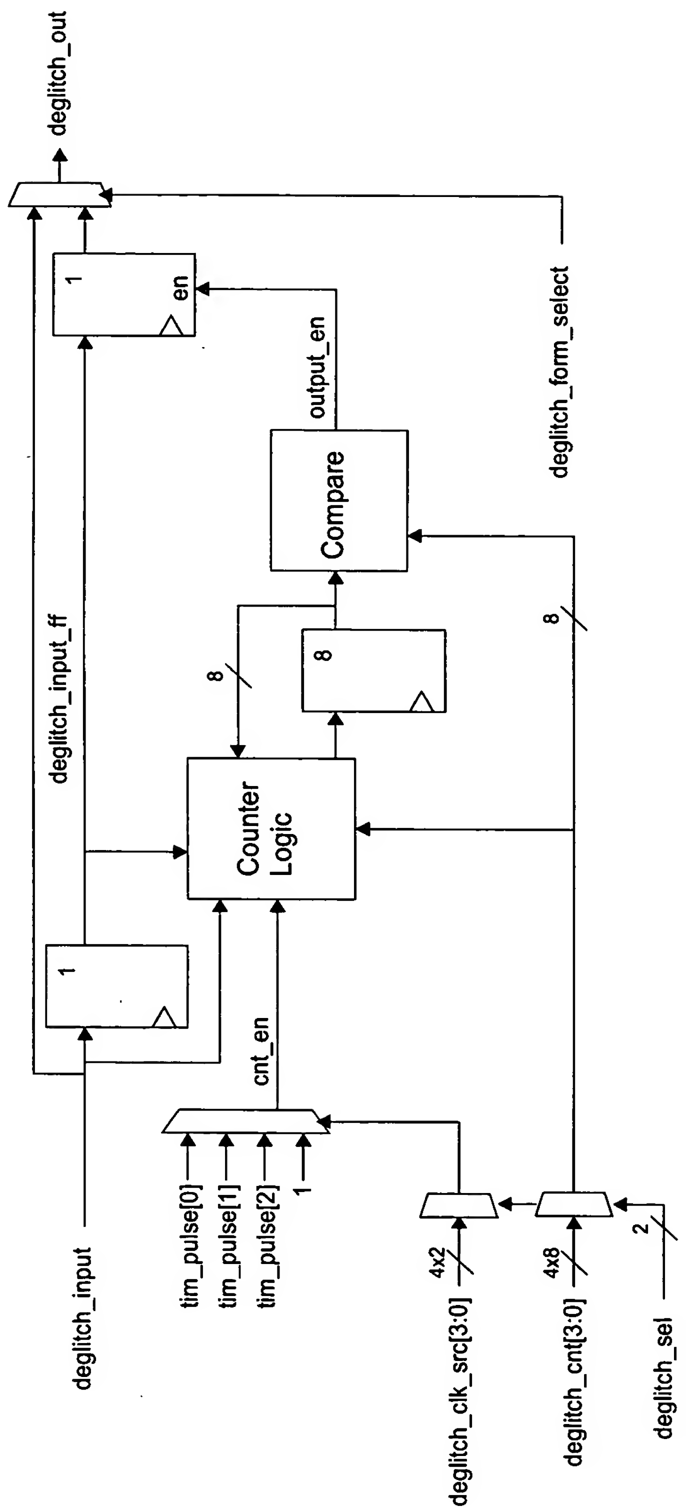

deglitch_out
deglitch_input_ff
deglitch_input
1
en
output_en
Compare
Counter
Logic
8
8
8
1
cnt_en
tim_pulse[0]
tim_pulse[1]
tim_pulse[2]
1
deglitch_form_select
deglitch_clk_src[3:0]
4x2
deglitch_cnt[3:0]
4x8
deglitch_sel
2

FIG. 57

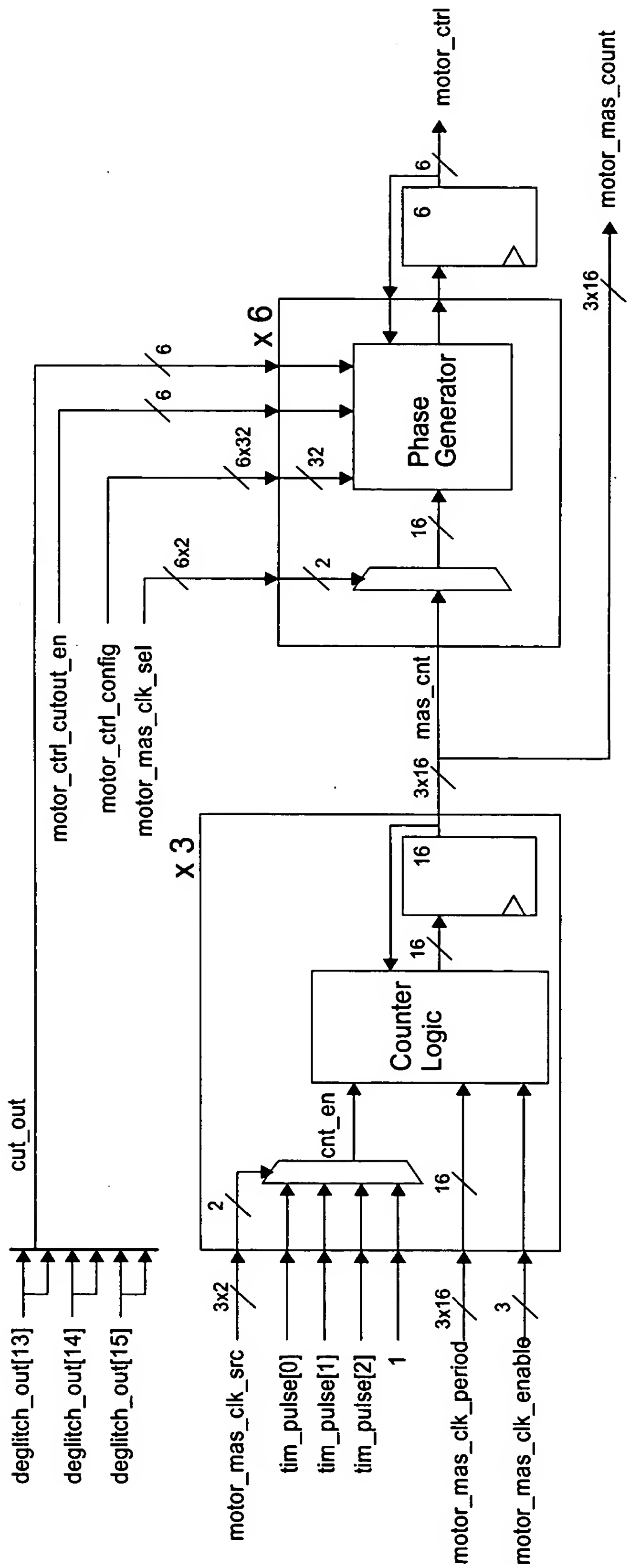

deglitch_out[13]
deglitch_out[14]
deglitch_out[15]
cut_out
motor_mas_clk_src
tim_pulse[0]
tim_pulse[1]
tim_pulse[2]
1
motor_mas_clk_period
motor_mas_clk_enable
3x2
3x16
3
2
16
cnt_en
Counter Logic
16
16
x 3
3x16
mas_cnt
motor_ctrl_cutout_en
motor_ctrl_config
motor_mas_clk_sel
6x2
2
16
6x32
32
6
6
Phase Generator
x 6
6
6
motor_ctrl
3x16
motor_mas_count

FIG. 58

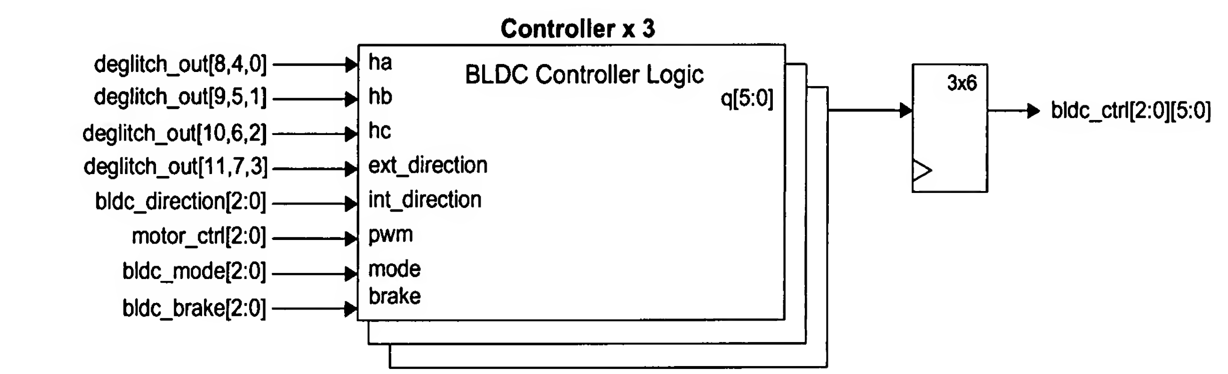

Controller x 3
BLDC Controller Logic
deglitch_out[8,4,0]
deglitch_out[9,5,1]
deglitch_out[10,6,2]
deglitch_out[11,7,3]
bldc_direction[2:0]
motor_ctrl[2:0]
bldc_mode[2:0]
bldc_brake[2:0]
ha
hb
hc
ext_direction
int_direction
pwm
mode
brake
q[5:0]
3x6
bldc_ctrl[2:0][5:0]

FIG. 59

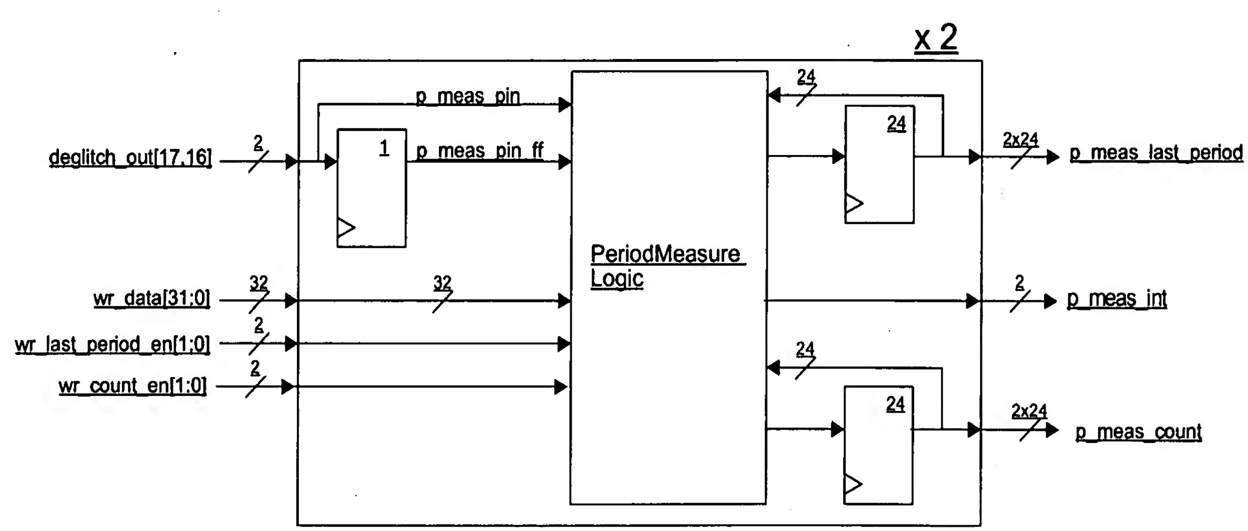

x 2
deglitch_out[17,16]
2
1
p_meas_pin
p_meas_pin_ff
wr_data[31:0]
32
32
wr_last_period_en[1:0]
2
wr_count_en[1:0]
2
PeriodMeasure Logic
24
24
24
24
2x24
p_meas_last_period
2
p_meas_int
2x24
p_meas_count

FIG. 60

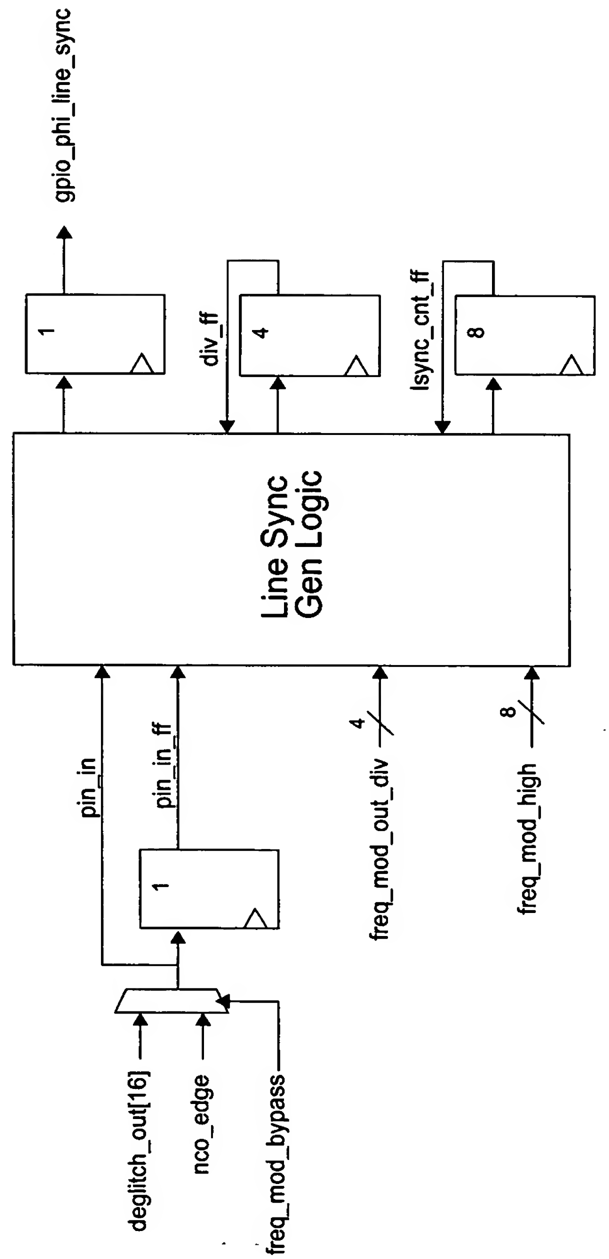

gpio_phi_line_sync
1
div_ff
4
lsync_cnt_ff
8
Line Sync
Gen Logic
pin_in
pin_in_ff
1
4
freq_mod_out_div
8
freq_mod_high
deglitch_out[16]
nco_edge
freq_mod_bypass

FIG. 61

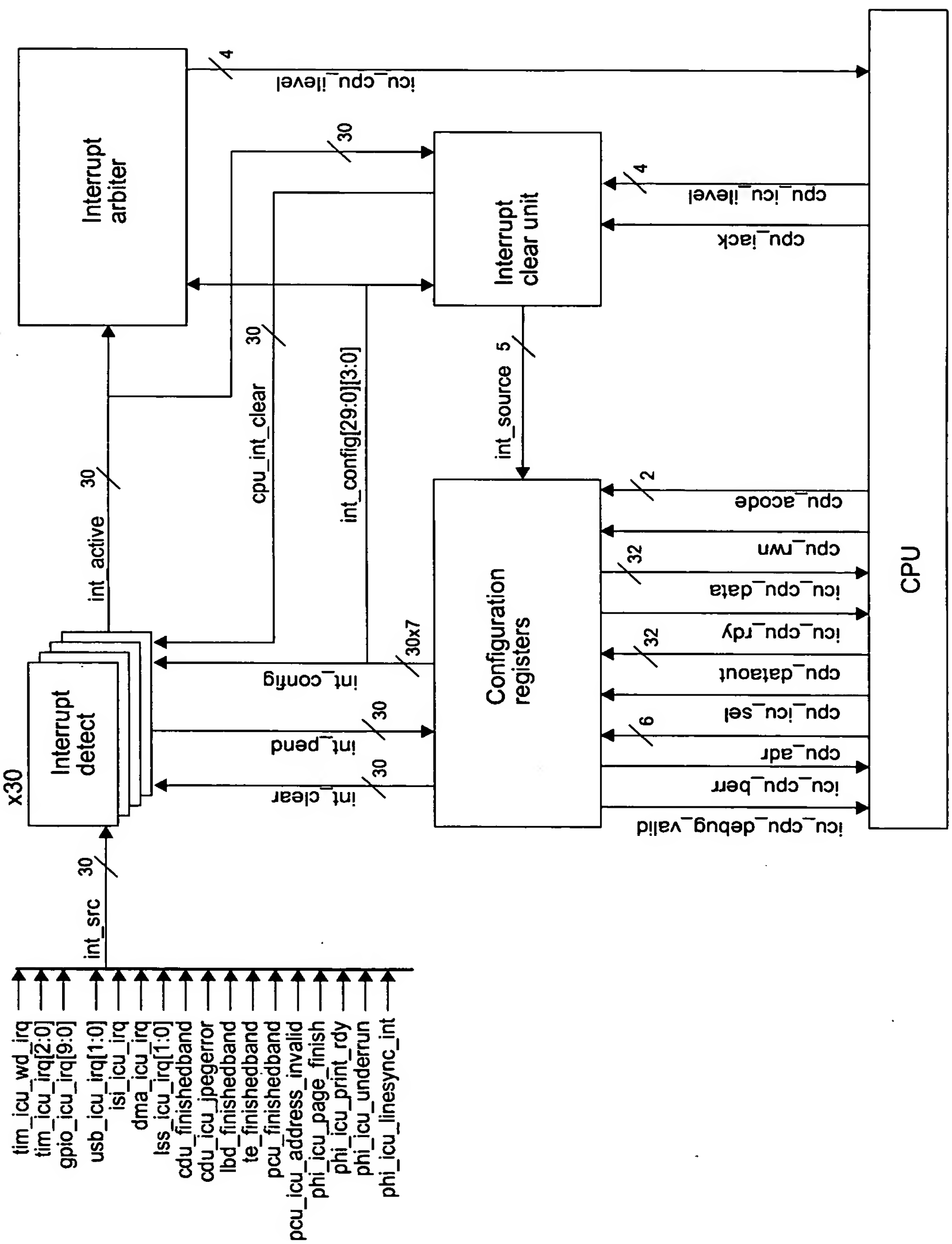
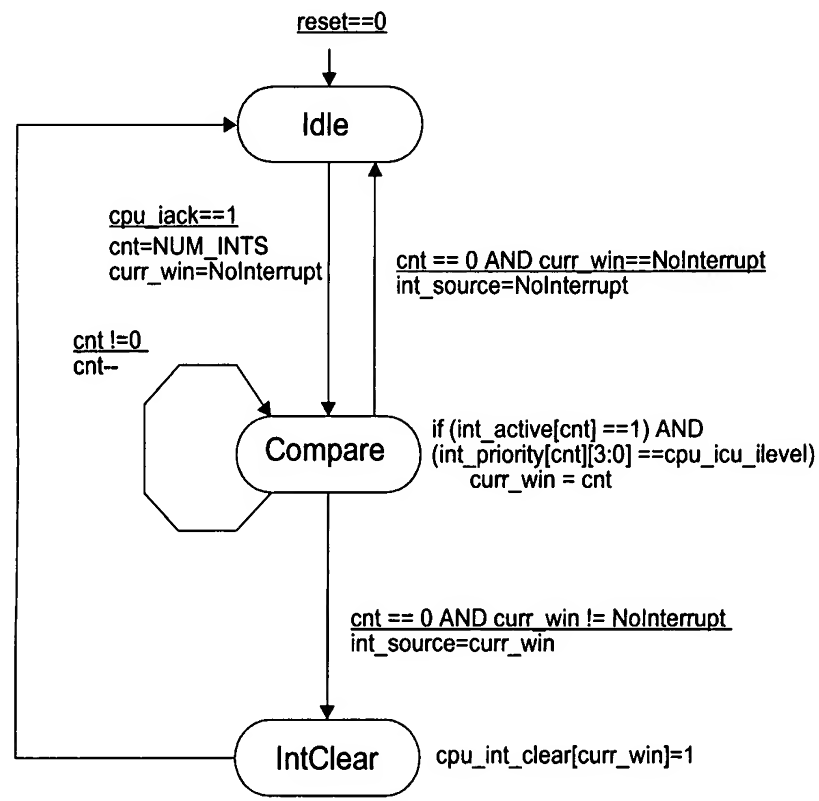

Interrupt arbiter
Interrupt clear unit
Configuration registers
Interrupt detect
x30
CPU
icu_cpu_ilevel
4
30
cpu_icu_ilevel
cpu_iack
int_source
5
30
cpu_int_clear
int_config[29:0][3:0]
int_active
30
30x7
int_config
int_pend
int_clear
cpu_acode
2
cpu_rwn
icu_cpu_data
32
icu_cpu_rdy
cpu_dataout
32
cpu_icu_sel
6
cpu_adr
icu_cpu_berr
icu_cpu_debug_valid
int_src
tim_icu_wd_irq
tim_icu_irq[2:0]
gpio_icu_irq[9:0]
usb_icu_irq[1:0]
isi_icu_irq
dma_icu_irq
lss_icu_irq[1:0]
cdu_finishedband
cdu_icu_jpegerror
lbd_finishedband
te_finishedband
pcu_finishedband
pcu_icu_address_invalid
phi_icu_page_finish
phi_icu_print_rdy
phi_icu_underrun
phi_icu_linesync_int

FIG. 62



Machine remains in same state by default
All outputs are zero unless otherwise stated

State Description:

Idle :	Normal Idle state
Compare:	Compare interrupt level. Determines the interrupt source
IntClear:	Interrupt clear, clear the pending bit for the current interrupt vector

FIG. 63

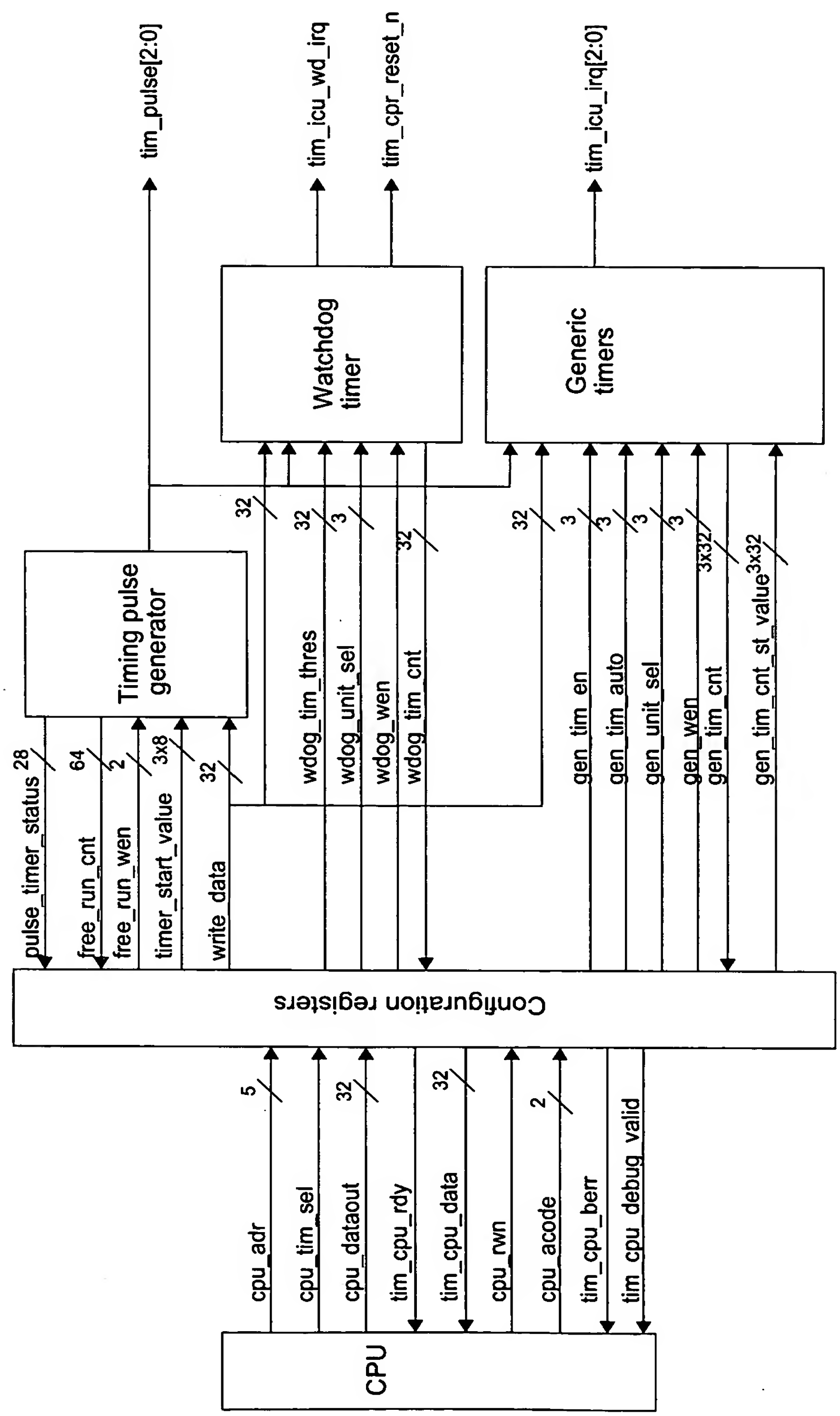

tim_pulse[2:0]
tim_icu_wd_irq
tim_cpr_reset_n
tim_icu_irq[2:0]
Watchdog timer
Generic timers
Timing pulse generator
32
3
3x32
3x8
28
64
2
5
pulse_timer_status
free_run_cnt
free_run_wen
timer_start_value
write data
wdog_tim_thres
wdog_unit_sel
wdog_wen
wdog_tim_cnt
gen_tim_en
gen_tim_auto
gen_unit_sel
gen_wen
gen_tim_cnt
gen_tim_cnt_st_value
Configuration registers
cpu_adr
cpu_tim_sel
cpu_dataout
tim_cpu_rdy
tim_cpu_data
cpu_rwn
cpu_acode
tim_cpu_berr
tim_cpu_debug_valid
CPU

FIG. 63A

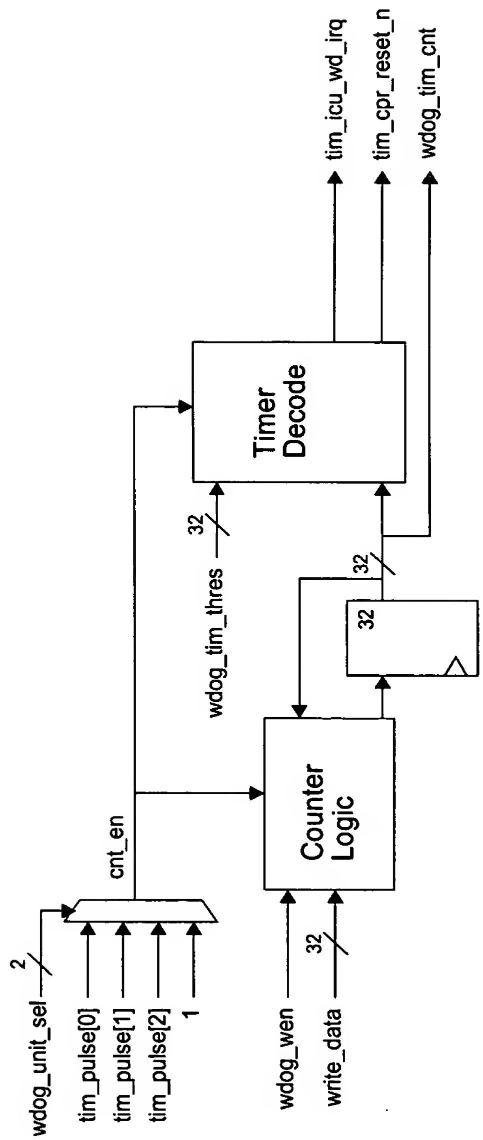

wdog_unit_sel
2
tim_pulse[0]
tim_pulse[1]
tim_pulse[2]
1
cnt_en
wdog_wen
write_data
32
Counter Logic
32
32
32
wdog_tim_thres
Timer Decode
tim_icu_wd_irq
tim_cpr_reset_n
wdog_tim_cnt

FIG. 64

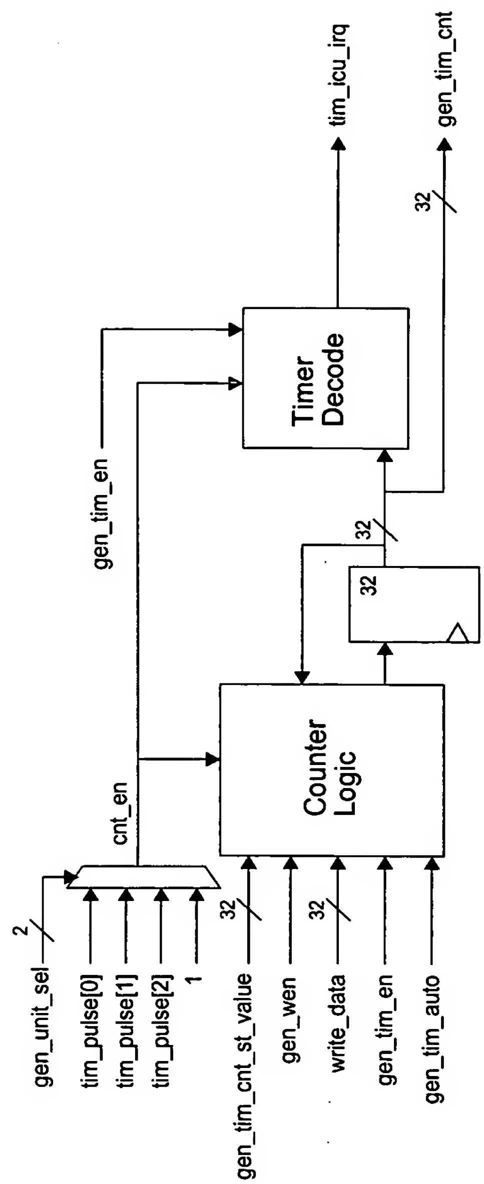

gen_unit_sel
2
tim_pulse[0]
tim_pulse[1]
tim_pulse[2]
1
cnt_en
gen_tim_cnt_st_value
32
gen_wen
write_data
32
gen_tim_en
gen_tim_auto
Counter Logic
32
32
gen_tim_en
Timer Decode
tim_icu_irq
32
gen_tim_cnt

FIG. 65

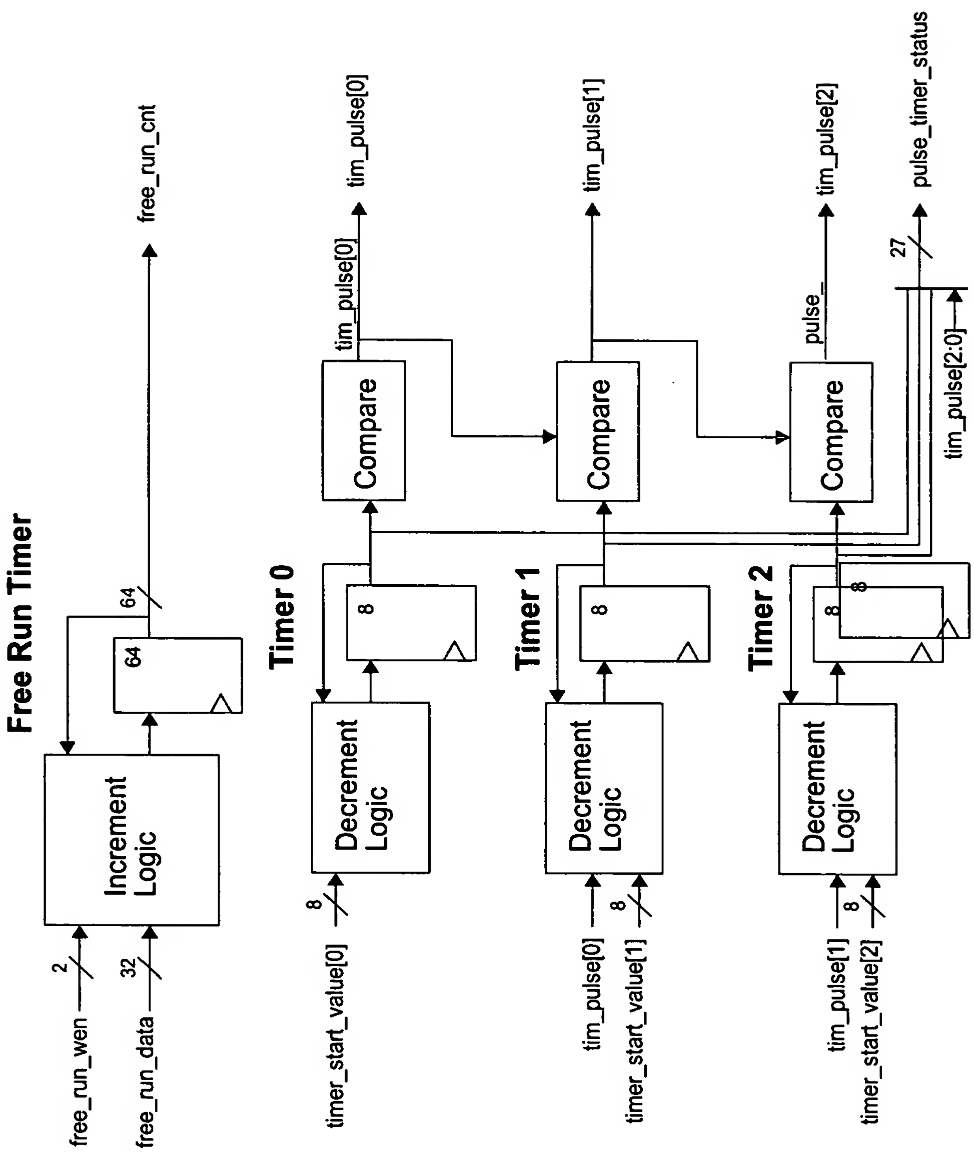

Free Run Timer
free_run_wen
2
free_run_data
32
Increment Logic
64
64
free_run_cnt
Timer 0
timer_start_value[0]
8
Decrement Logic
8
Compare
tim_pulse[0]
tim_pulse[0]
Timer 1
tim_pulse[0]
timer_start_value[1]
8
Decrement Logic
8
Compare
tim_pulse[1]
Timer 2
tim_pulse[1]
timer_start_value[2]
8
Decrement Logic
8
8
Compare
pulse_
tim_pulse[2]
27
pulse_timer_status
tim_pulse[2:0]

FIG. 66

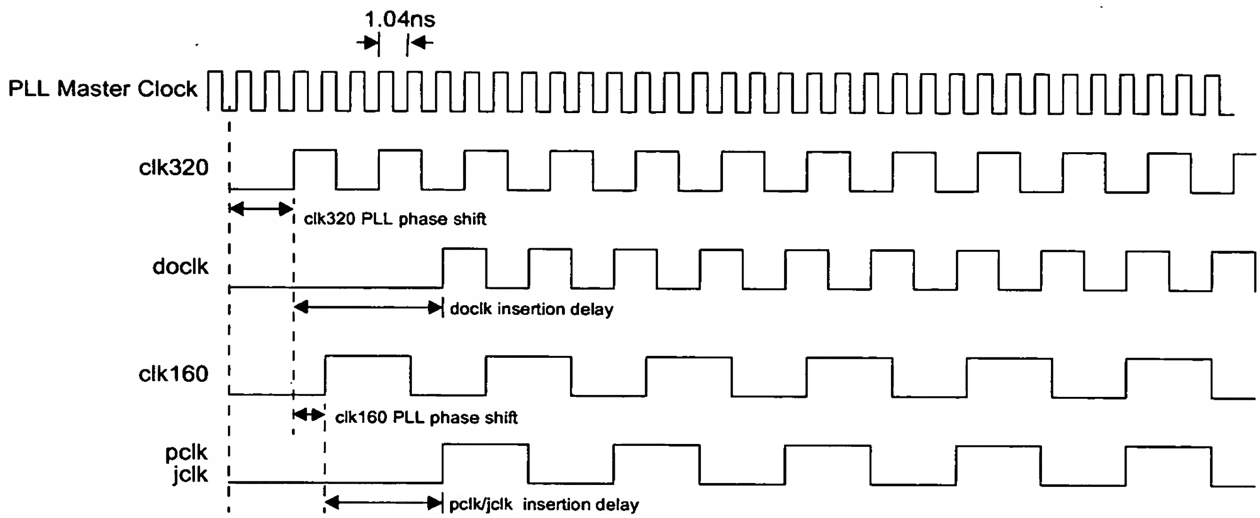

1.04ns
PLL Master Clock
clk320
clk320 PLL phase shift
doclk
doclk insertion delay
clk160
clk160 PLL phase shift
pclk
jclk
pclk/jclk insertion delay

FIG. 67

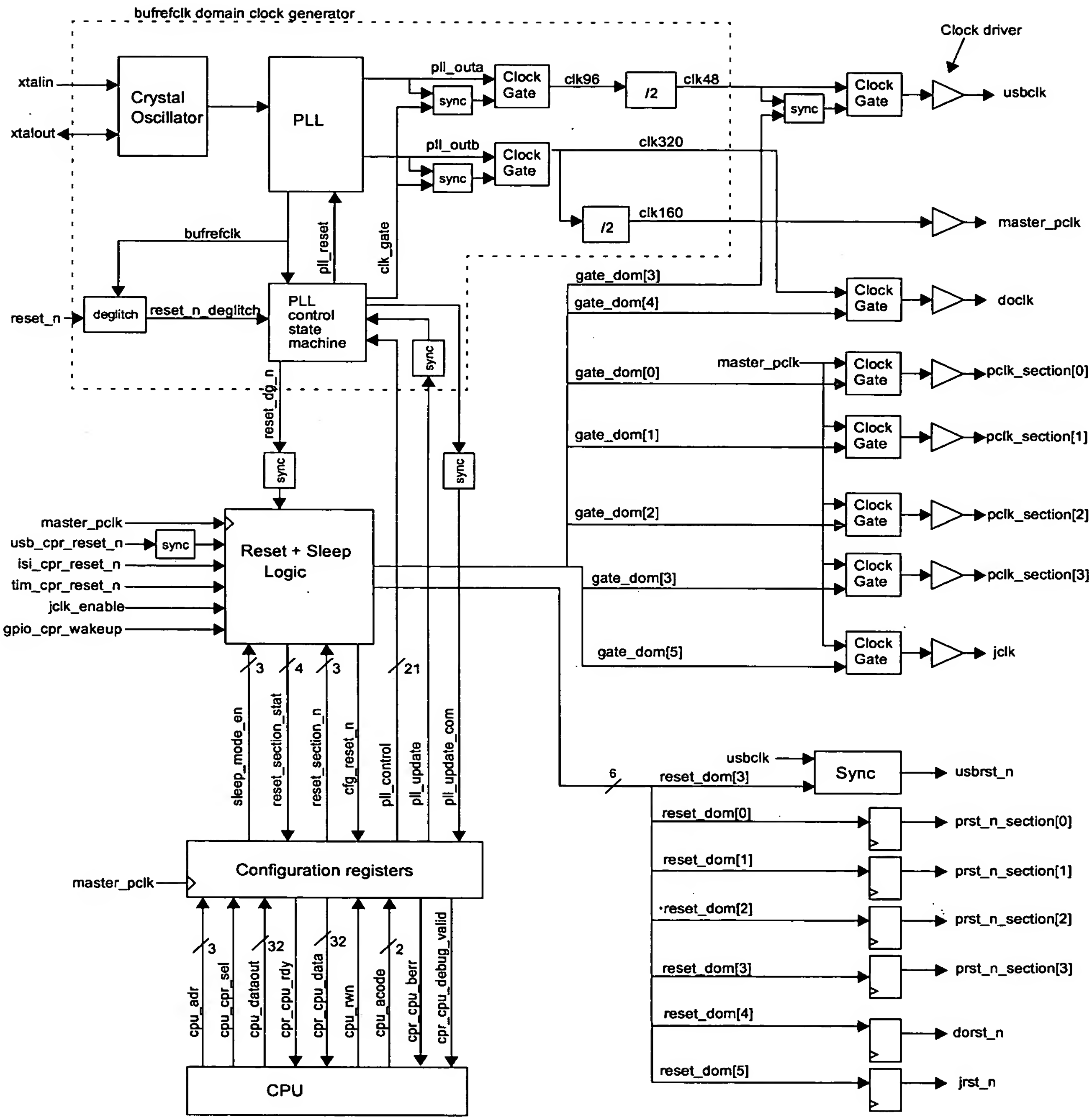

bufrefclk domain clock generator
xtalin
xtalout
Crystal Oscillator
PLL
pll_outa
pll_outb
sync
Clock Gate
clk96
/2
clk48
clk320
clk160
Clock driver
usbclk
master_pclk
doclk
bufrefclk
pll_reset
clk_gate
reset_n
deglitch
reset_n_deglitch
PLL control state machine
reset_dg_n
gate_dom[3]
gate_dom[4]
gate_dom[0]
gate_dom[1]
gate_dom[2]
gate_dom[5]
pclk_section[0]
pclk_section[1]
pclk_section[2]
pclk_section[3]
jclk
usb_cpr_reset_n
isi_cpr_reset_n
tim_cpr_reset_n
jclk_enable
gpio_cpr_wakeup
Reset + Sleep Logic
sleep_mode_en
reset_section_stat
reset_section_n
cfg_reset_n
pll_control
pll_update
pll_update_cmd
Configuration registers
cpu_adr
cpu_cpr_sel
cpu_dataout
cpr_cpu_rdy
cpr_cpu_data
cpu_rwn
cpu_acode
cpr_cpu_berr
cpr_cpu_debug_valid
CPU
Sync
usbrst_n
reset_dom[3]
reset_dom[0]
reset_dom[1]
reset_dom[2]
reset_dom[4]
reset_dom[5]
prst_n_section[0]
prst_n_section[1]
prst_n_section[2]
prst_n_section[3]
dorst_n
jrst_n

FIG. 68

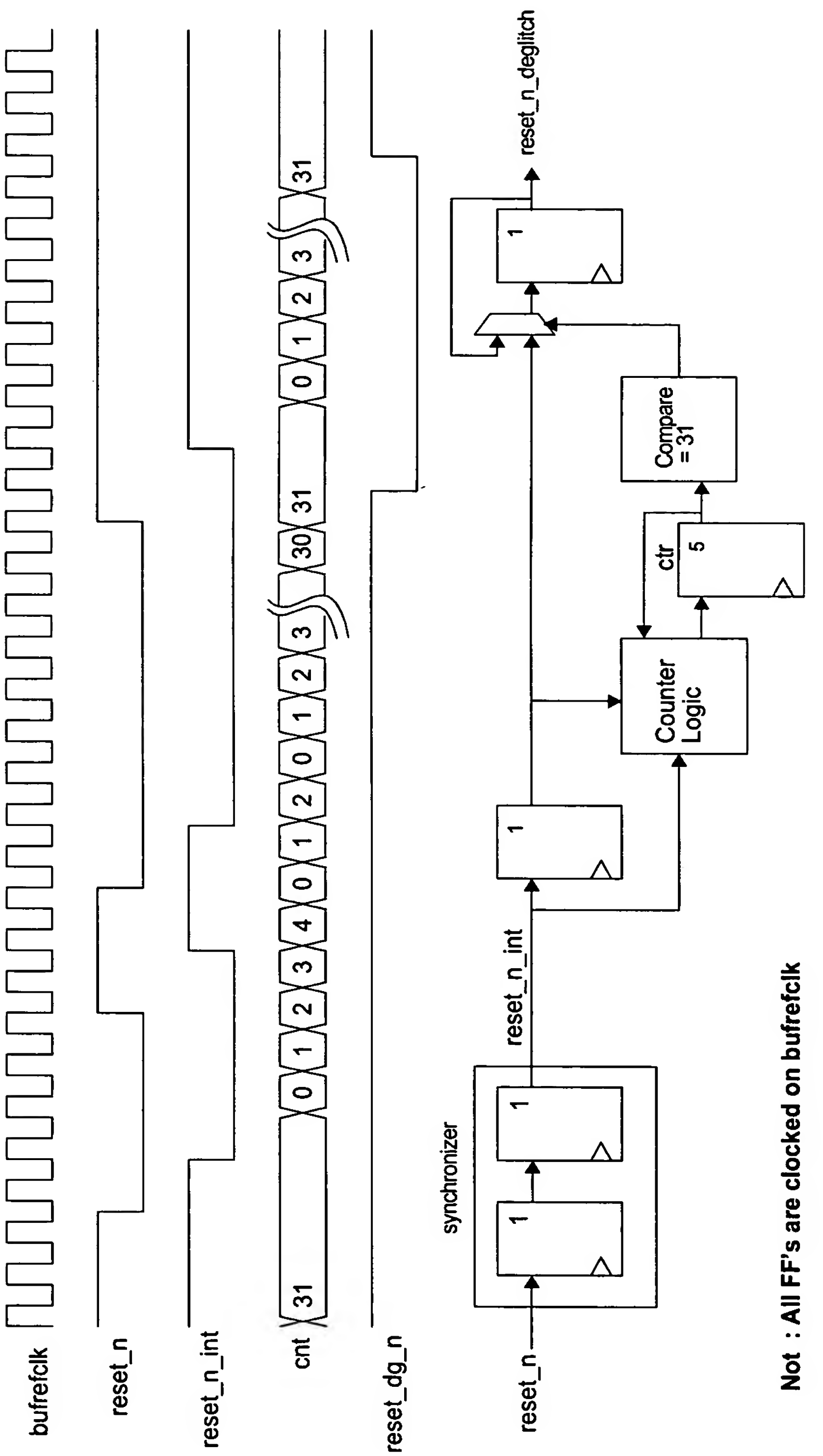

bufrefclk
reset_n
reset_n_int
cnt
31
0
1
2
3
4
0
1
2
0
1
2
3
30
31
0
1
2
3
31
reset_dg_n
synchronizer
reset_n
1
1
reset_n_int
1
Counter Logic
ctr
5
Compare = 31
1
reset_n_deglitch
Not : All FF's are clocked on bufrefclk

FIG. 69

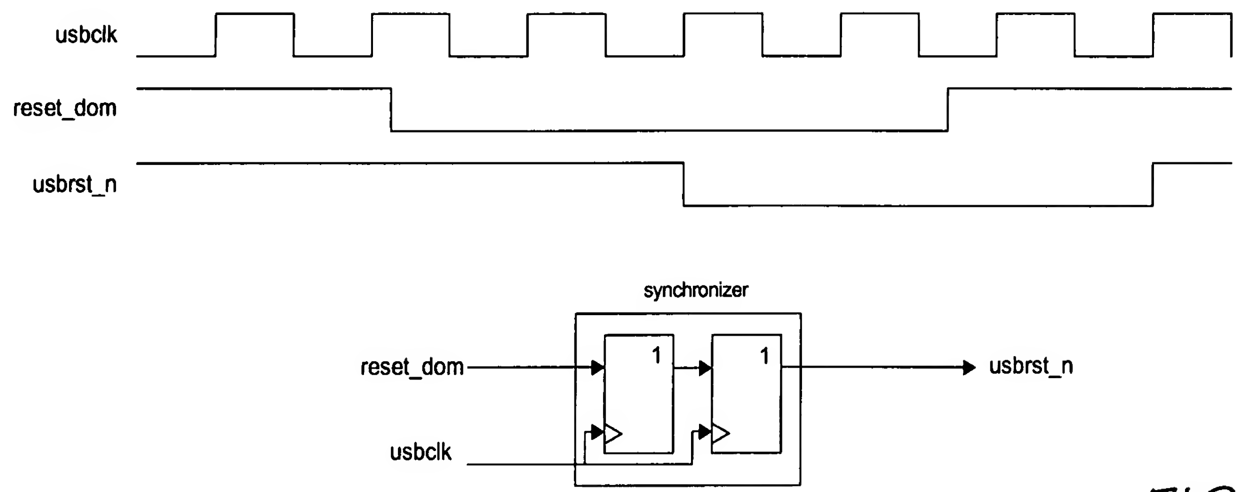

usbclk
reset_dom
usbrst_n
synchronizer
reset_dom
usbclk
1
1
usbrst_n

FIG. 70

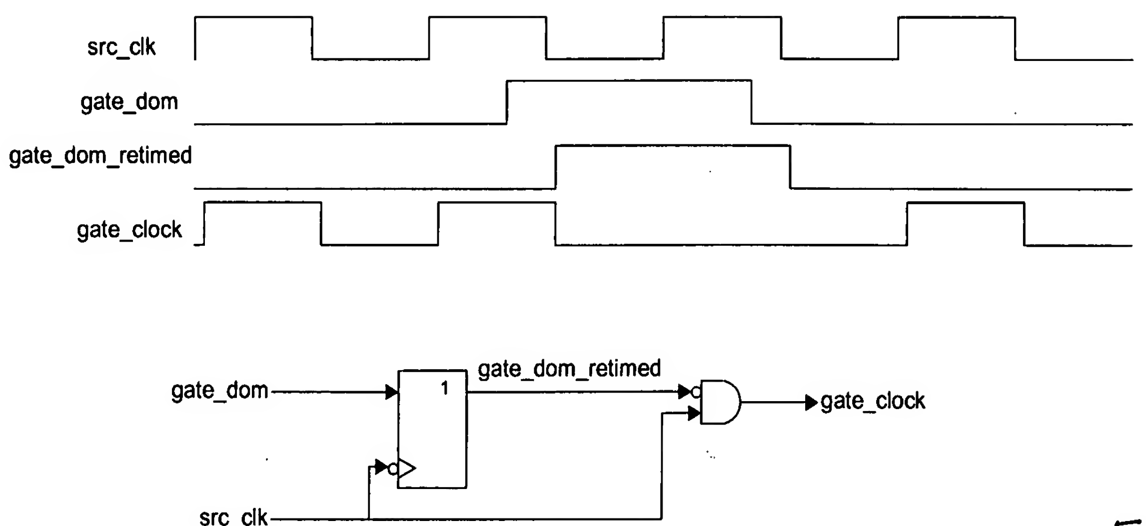
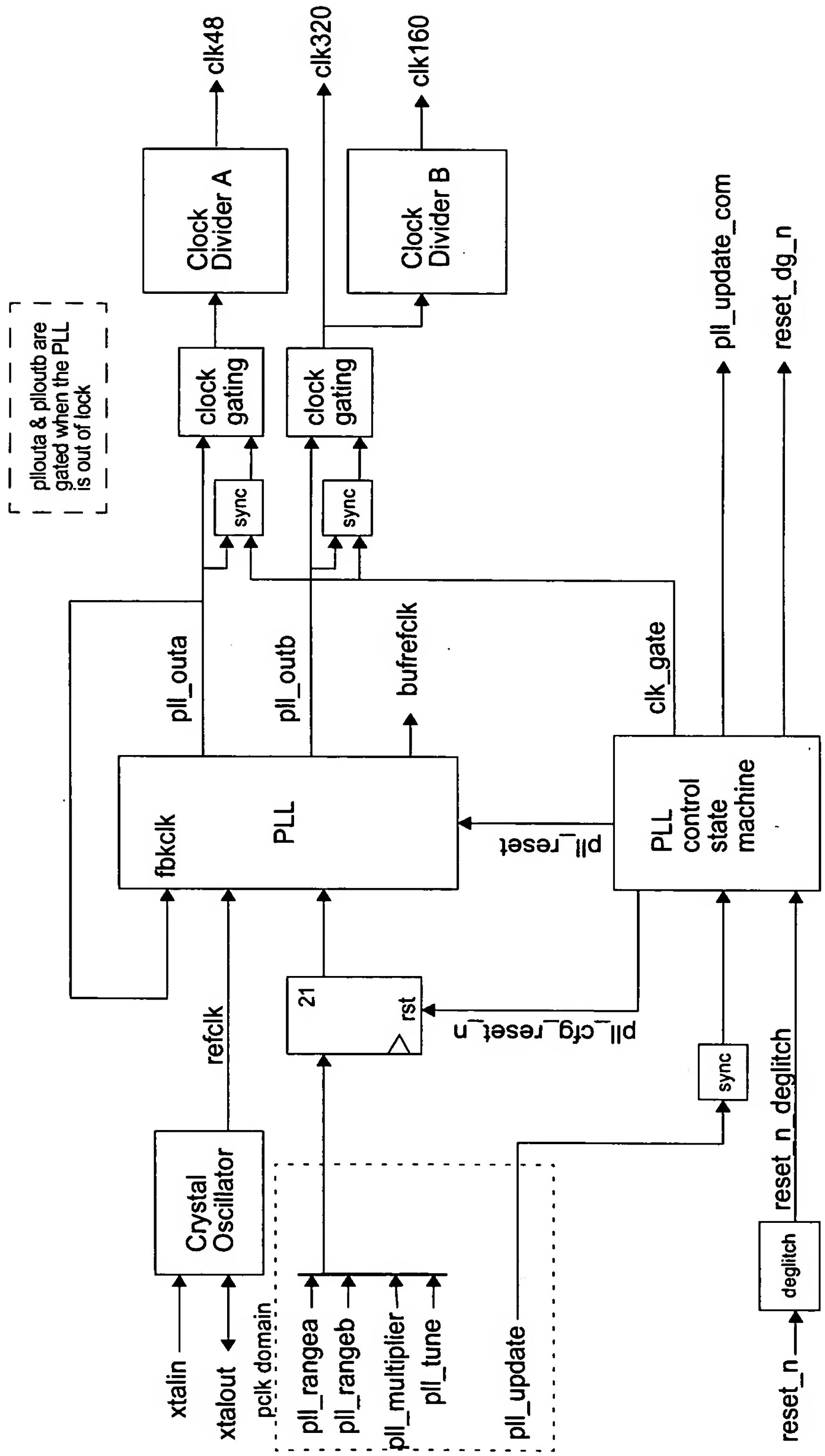

src_clk
gate_dom
gate_dom_retimed
gate_clock
gate_dom
src_clk
1
gate_dom_retimed
gate_clock

FIG. 71



Note: All logic clocked on bufrefclk unless otherwise indicated

FIG. 72

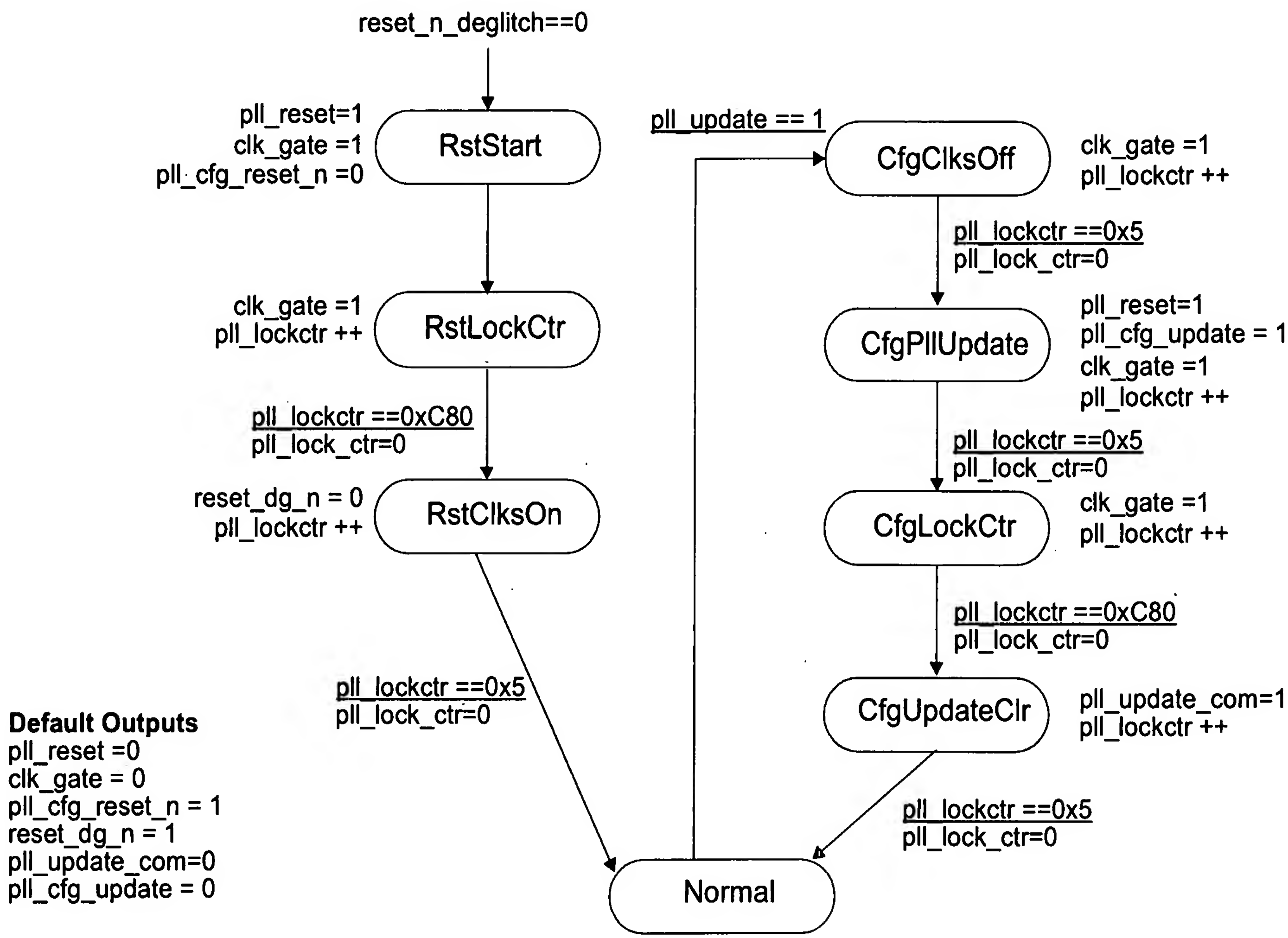

reset_n_deglitch==0
pll_reset=1
clk_gate =1
pll_cfg_reset_n =0
RstStart
clk_gate =1
pll_lockctr ++
RstLockCtr
pll_lockctr ==0xC80
pll_lock_ctr=0
reset_dg_n = 0
pll_lockctr ++
RstClksOn
pll_lockctr ==0x5
pll_lock_ctr=0
Normal
pll_update == 1
CfgClksOff
clk_gate =1
pll_lockctr ++
pll_lockctr ==0x5
pll_lock_ctr=0
CfgPllUpdate
pll_reset=1
pll_cfg_update = 1
clk_gate =1
pll_lockctr ++
pll_lockctr ==0x5
pll_lock_ctr=0
CfgLockCtr
clk_gate =1
pll_lockctr ++
pll_lockctr ==0xC80
pll_lock_ctr=0
CfgUpdateClr
pll_update_com=1
pll_lockctr ++
pll_lockctr ==0x5
pll_lock_ctr=0
Default Outputs
pll_reset =0
clk_gate = 0
pll_cfg_reset_n = 1
reset_dg_n = 1
pll_update_com=0
pll_cfg_update = 0

FIG. 73

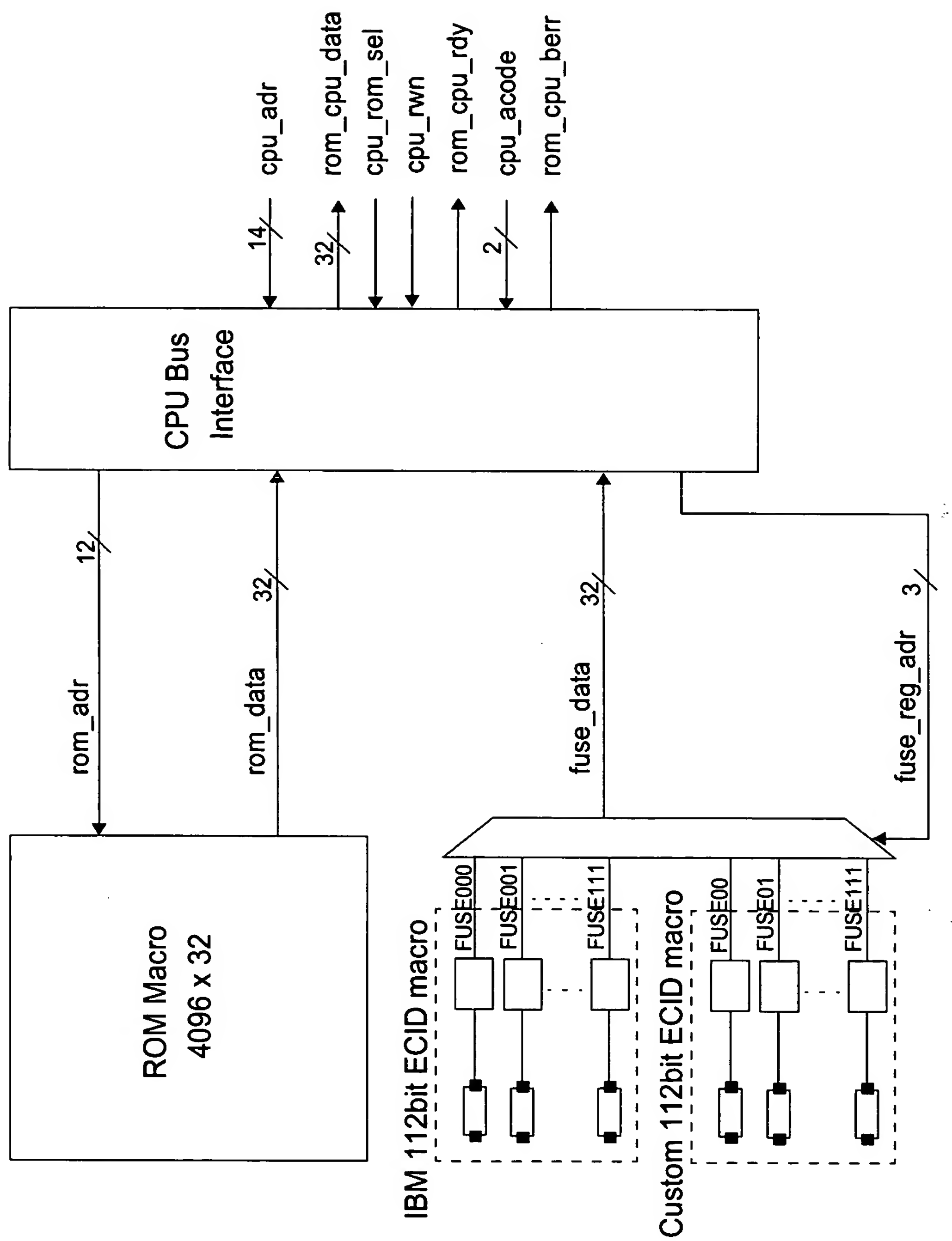

cpu_adr
rom_cpu_data
cpu_rom_sel
cpu_rwn
rom_cpu_rdy
cpu_acode
rom_cpu_berr
14
32
2
CPU Bus
Interface
12
32
32
3
rom_adr
rom_data
fuse_data
fuse_reg_adr
ROM Macro
4096 x 32
FUSE000
FUSE001
FUSE111
FUSE00
FUSE01
FUSE111
IBM 112bit ECID macro
Custom 112bit ECID macro

FIG. 74

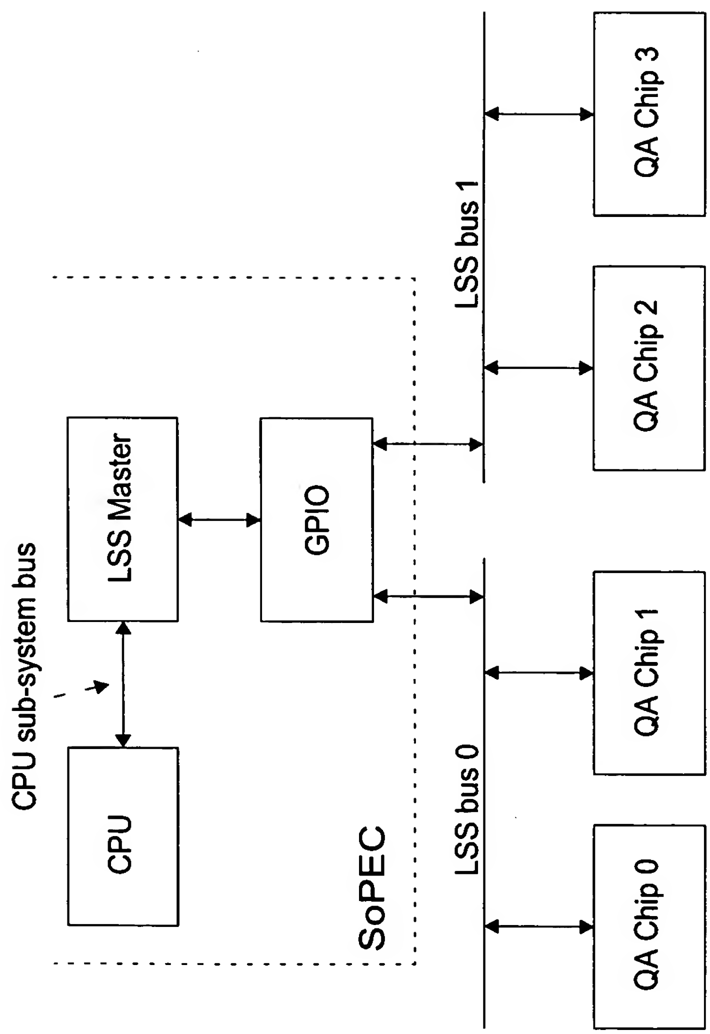

CPU sub-system bus
CPU
LSS Master
GPIO
SoPEC
LSS bus 0
LSS bus 1
QA Chip 0
QA Chip 1
QA Chip 2
QA Chip 3

FIG. 75

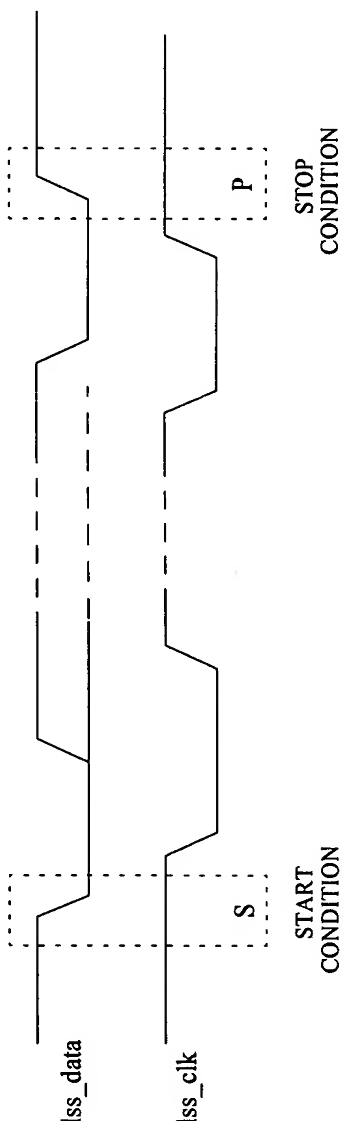

lss_data
lss_clk
S
START
CONDITION
P
STOP
CONDITION

FIG. 76

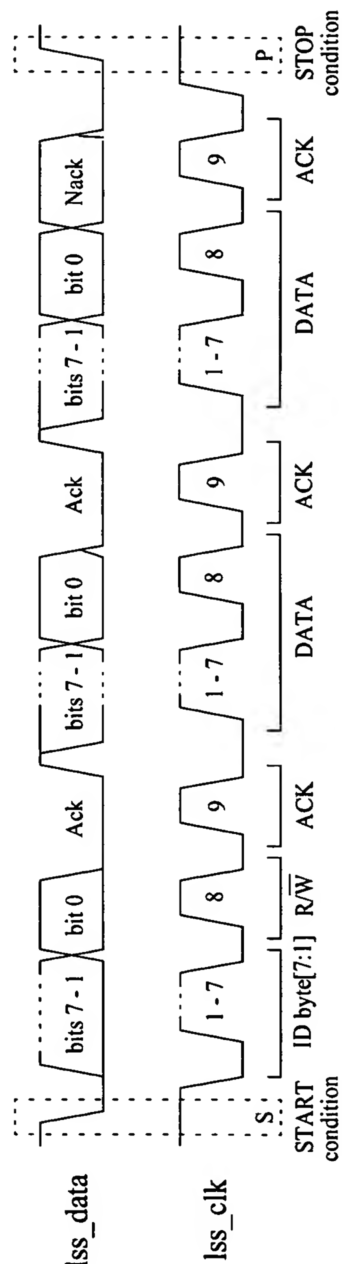

lss_data
lss_clk
bits 7 - 1
bit 0
Ack
bits 7 - 1
bit 0
Ack
bits 7 - 1
bit 0
Nack
1 - 7
8
9
1 - 7
8
9
1 - 7
8
9
S
START
condition
ID byte[7:1]
R/W̄
ACK
DATA
ACK
DATA
ACK
P
STOP
condition

FIG. 77

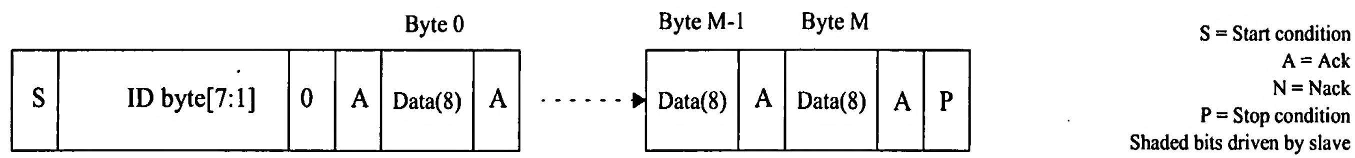

Byte 0
Byte M-1
Byte M
S
ID byte[7:1]
0
A
Data(8)
A
Data(8)
A
Data(8)
A
P
S = Start condition
A = Ack
N = Nack
P = Stop condition
Shaded bits driven by slave

FIG. 78

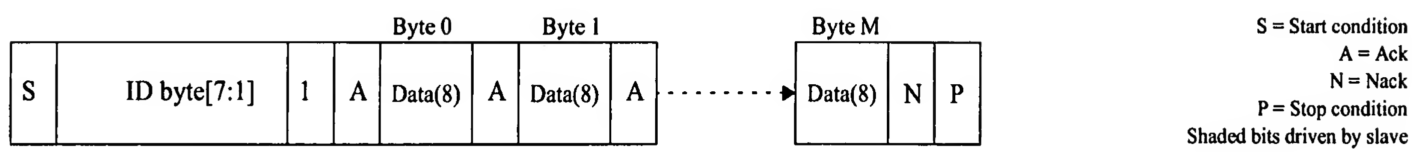

Byte 0
Byte 1
Byte M
S
ID byte[7:1]
1
A
Data(8)
A
Data(8)
A
Data(8)
N
P
S = Start condition
A = Ack
N = Nack
P = Stop condition
Shaded bits driven by slave

FIG. 79

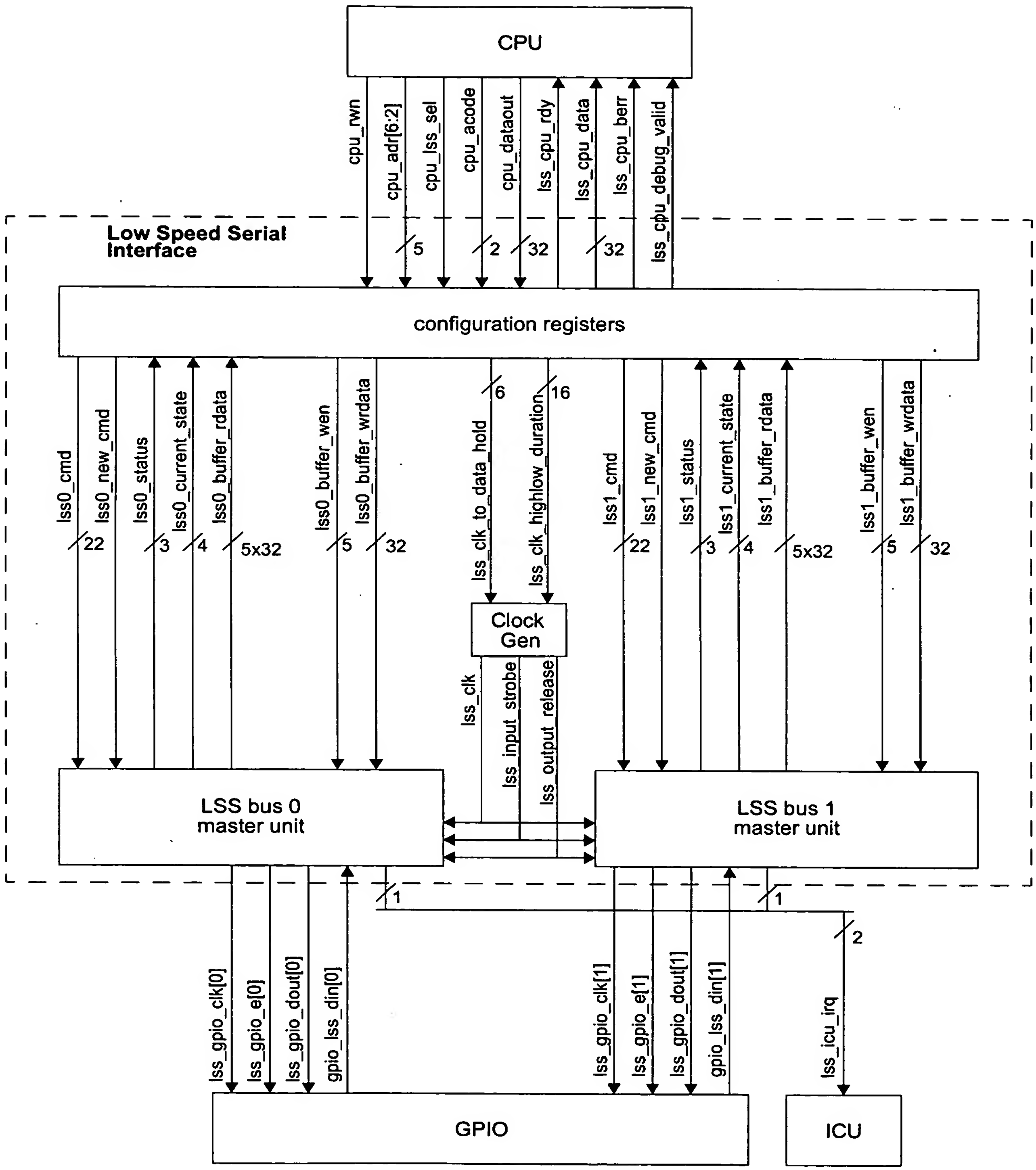

CPU
cpu_rwn
cpu_adr[6:2]
cpu_lss_sel
cpu_acode
cpu_dataout
lss_cpu_rdy
lss_cpu_data
lss_cpu_berr
lss_cpu_debug_valid
5
2
32
32
Low Speed Serial Interface
configuration registers
lss0_cmd
lss0_new_cmd
lss0_status
lss0_current_state
lss0_buffer_rdata
lss0_buffer_wen
lss0_buffer_wrdata
22
3
4
5x32
5
32
lss_clk_to_data_hold
lss_clk_highlow_duration
6
16
Clock Gen
lss_clk
lss_input_strobe
lss_output_release
lss1_cmd
lss1_new_cmd
lss1_status
lss1_current_state
lss1_buffer_rdata
lss1_buffer_wen
lss1_buffer_wrdata
22
3
4
5x32
5
32
LSS bus 0 master unit
LSS bus 1 master unit
1
1
2
lss_gpio_clk[0]
lss_gpio_e[0]
lss_gpio_dout[0]
gpio_lss_din[0]
lss_gpio_clk[1]
lss_gpio_e[1]
lss_gpio_dout[1]
gpio_lss_din[1]
lss_icu_irq
GPIO
ICU

FIG. 80

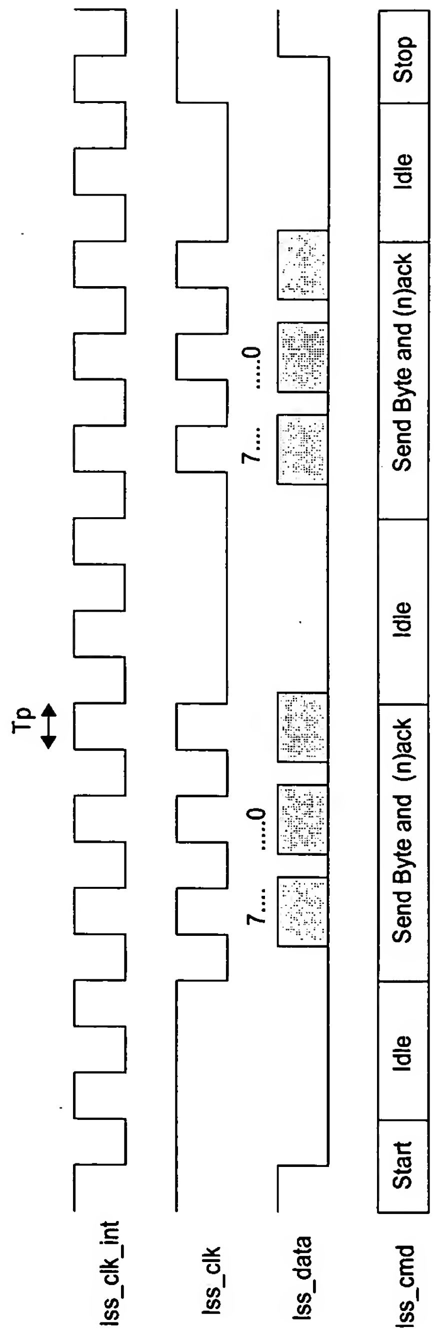

Tp
lss_clk_int
lss_clk
lss_data
7....
....0
7....
....0
lss_cmd
Start
Idle
Send Byte and (n)ack
Idle
Send Byte and (n)ack
Idle
Stop

FIG. 81

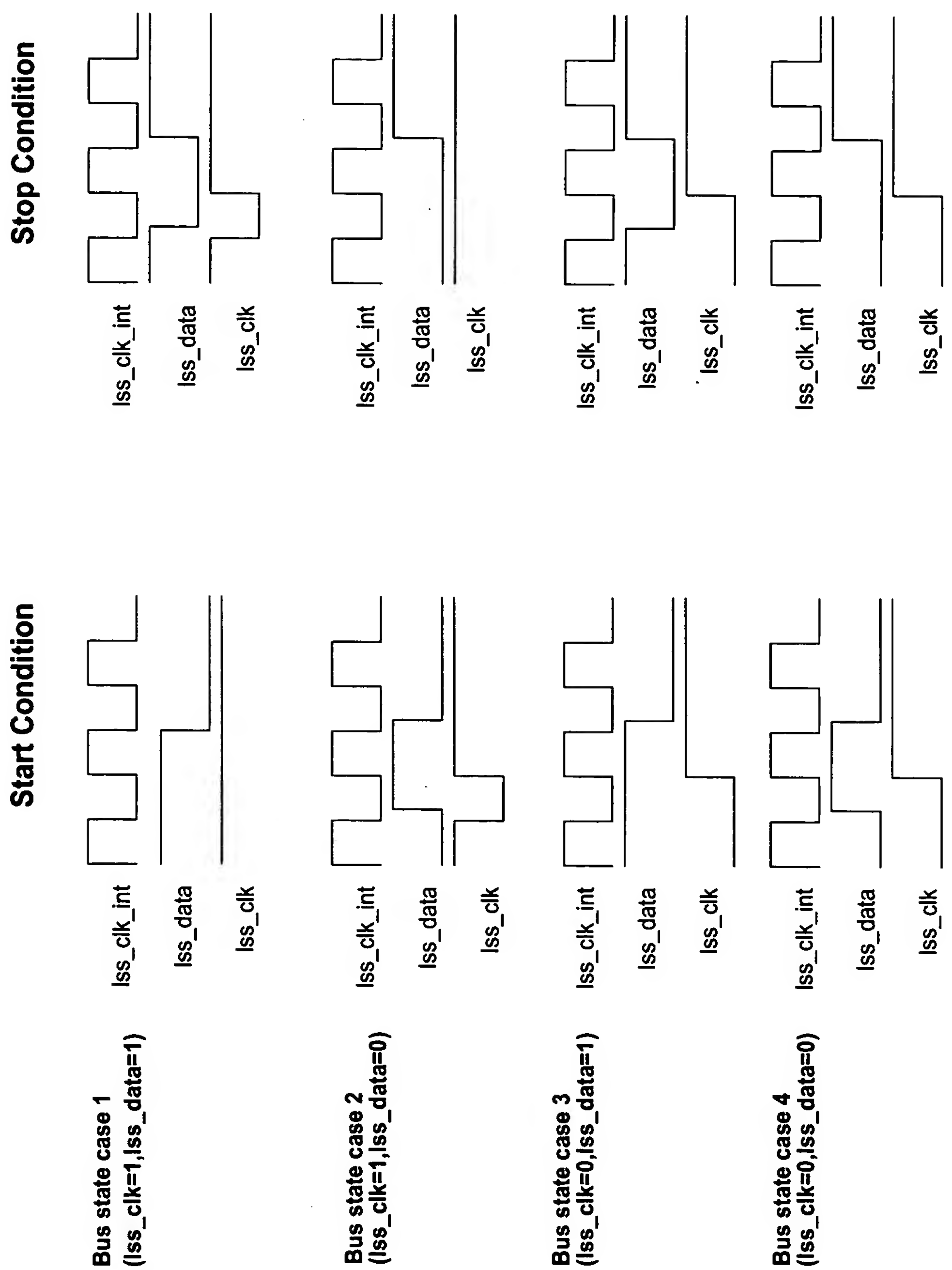
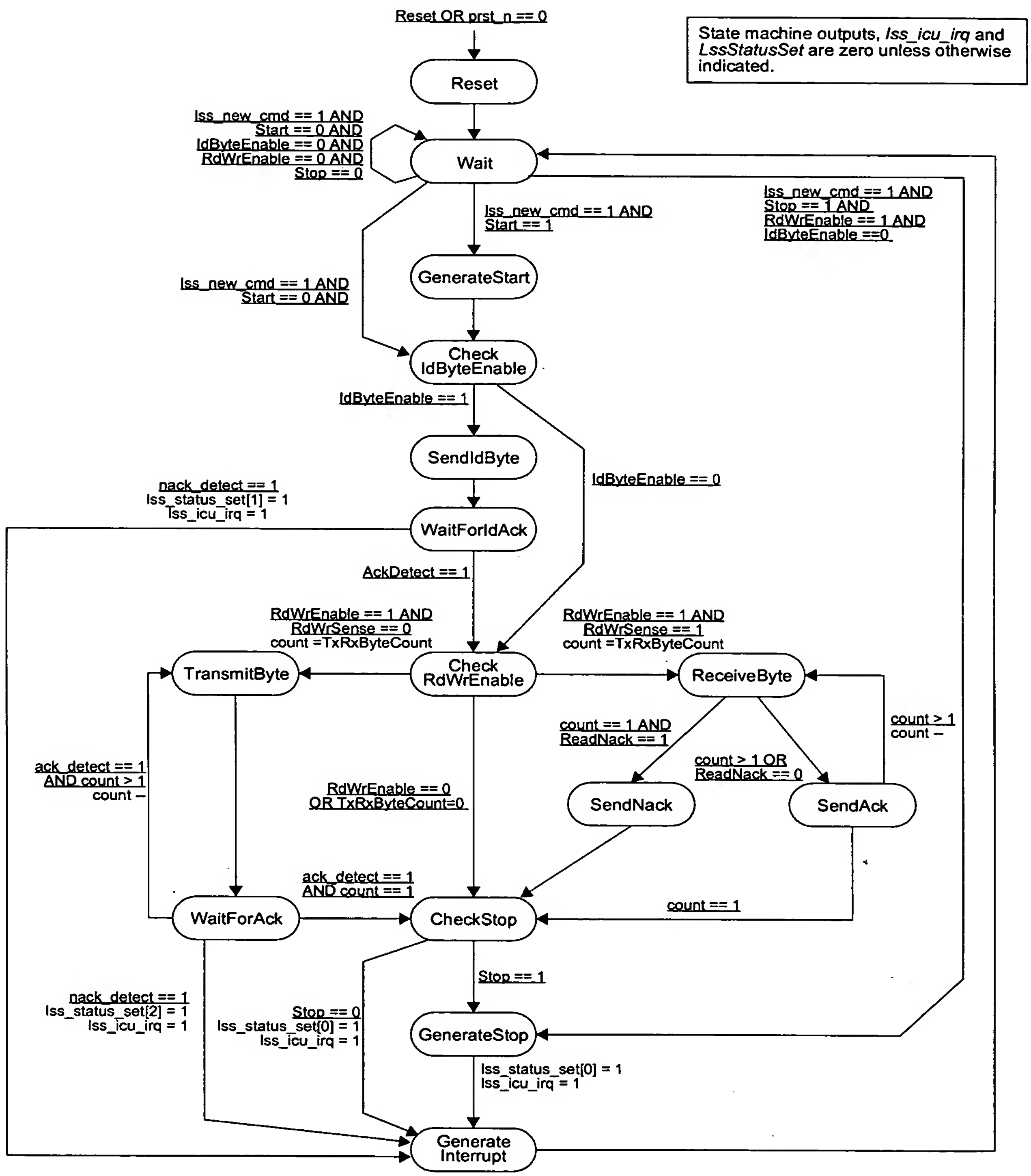

Start Condition
Stop Condition
Bus state case 1
(lss_clk=1,lss_data=1)
Bus state case 2
(lss_clk=1,lss_data=0)
Bus state case 3
(lss_clk=0,lss_data=1)
Bus state case 4
(lss_clk=0,lss_data=0)
lss_clk_int
lss_data
lss_clk

FIG. 82



Reset OR prst_n == 0
State machine outputs, Iss_icu_irq and LssStatusSet are zero unless otherwise indicated.
Reset
Iss_new_cmd == 1 AND
Start == 0 AND
IdByteEnable == 0 AND
RdWrEnable == 0 AND
Stop == 0
Wait
Iss_new_cmd == 1 AND
Stop == 1 AND
RdWrEnable == 1 AND
IdByteEnable ==0
Iss_new_cmd == 1 AND
Start == 1
GenerateStart
Iss_new_cmd == 1 AND
Start == 0 AND
Check IdByteEnable
IdByteEnable == 1
IdByteEnable == 0
SendIdByte
nack_detect == 1
Iss_status_set[1] = 1
Iss_icu_irq = 1
WaitForIdAck
AckDetect == 1
RdWrEnable == 1 AND
RdWrSense == 0
count =TxRxByteCount
RdWrEnable == 1 AND
RdWrSense == 1
count =TxRxByteCount
TransmitByte
Check RdWrEnable
ReceiveByte
count == 1 AND
ReadNack == 1
count > 1
count --
ack_detect == 1
AND count > 1
count --
RdWrEnable == 0
OR TxRxByteCount=0
count > 1 OR
ReadNack == 0
SendNack
SendAck
ack_detect == 1
AND count == 1
WaitForAck
CheckStop
count == 1
Stop == 1
nack_detect == 1
Iss_status_set[2] = 1
Iss_icu_irq = 1
Stop == 0
Iss_status_set[0] = 1
Iss_icu_irq = 1
GenerateStop
Iss_status_set[0] = 1
Iss_icu_irq = 1
Generate Interrupt

FIG. 83

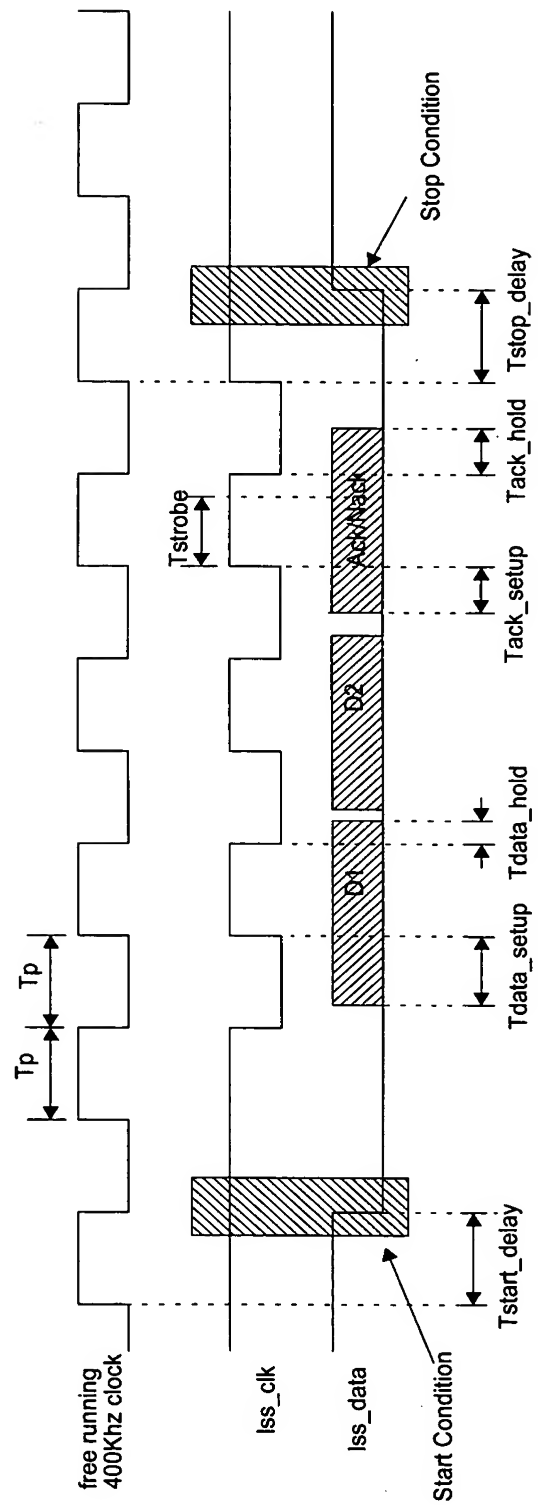

free running 400Khz clock
Iss_clk
Iss_data
Start Condition
Stop Condition
Tp
Tp
Tstrobe
D1
D2
Ack/Nack
Tstart_delay
Tdata_setup
Tdata_hold
Tack_setup
Tack_hold
Tstop_delay

FIG. 84

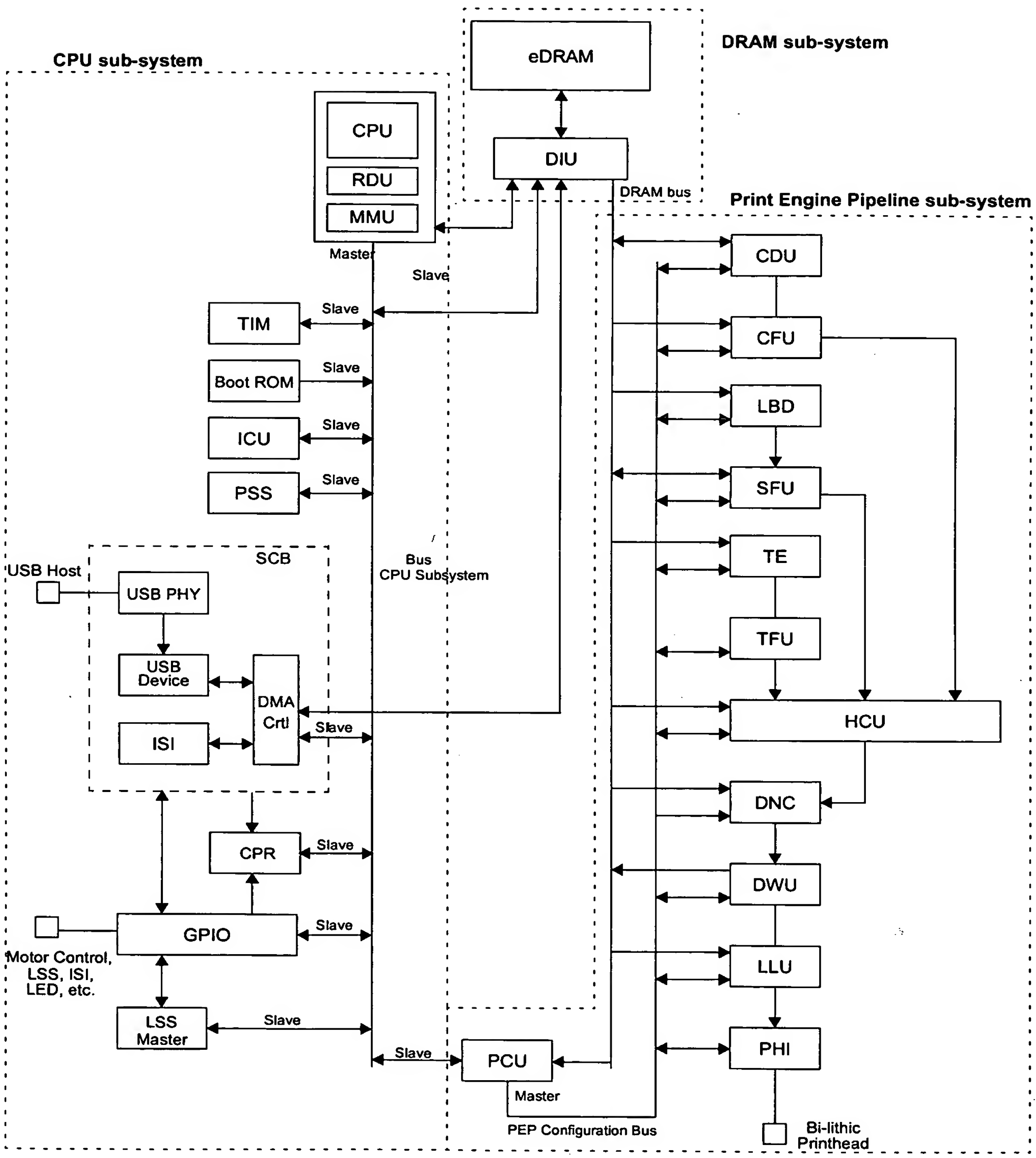

DRAM sub-system
eDRAM
DIU
DRAM bus
CPU sub-system
CPU
RDU
MMU
Master
Slave
TIM
Slave
Boot ROM
Slave
ICU
Slave
PSS
Slave
SCB
USB Host
USB PHY
USB Device
DMA Crtl
ISI
Slave
Bus
CPU Subsystem
CPR
Slave
GPIO
Slave
Motor Control, LSS, ISI, LED, etc.
LSS Master
Slave
Slave
PCU
Master
PEP Configuration Bus
Print Engine Pipeline sub-system
CDU
CFU
LBD
SFU
TE
TFU
HCU
DNC
DWU
LLU
PHI
Bi-lithic Printhead

FIG. 85

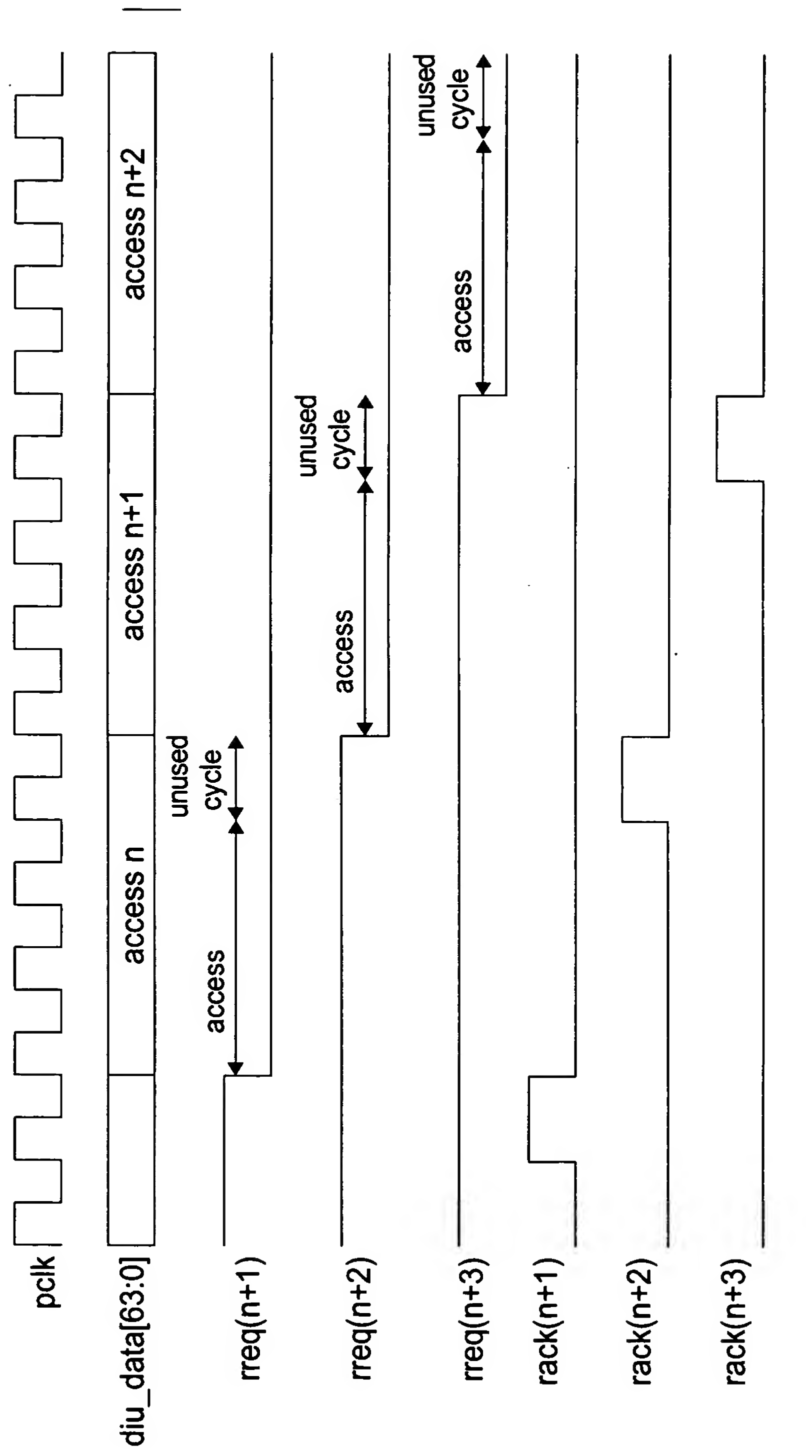

pclk
diu_data[63:0]
rreq(n+1)
rreq(n+2)
rreq(n+3)
rack(n+1)
rack(n+2)
rack(n+3)
access n
access n+1
access n+2
access
unused cycle
access
unused cycle
access
unused cycle

FIG. 86

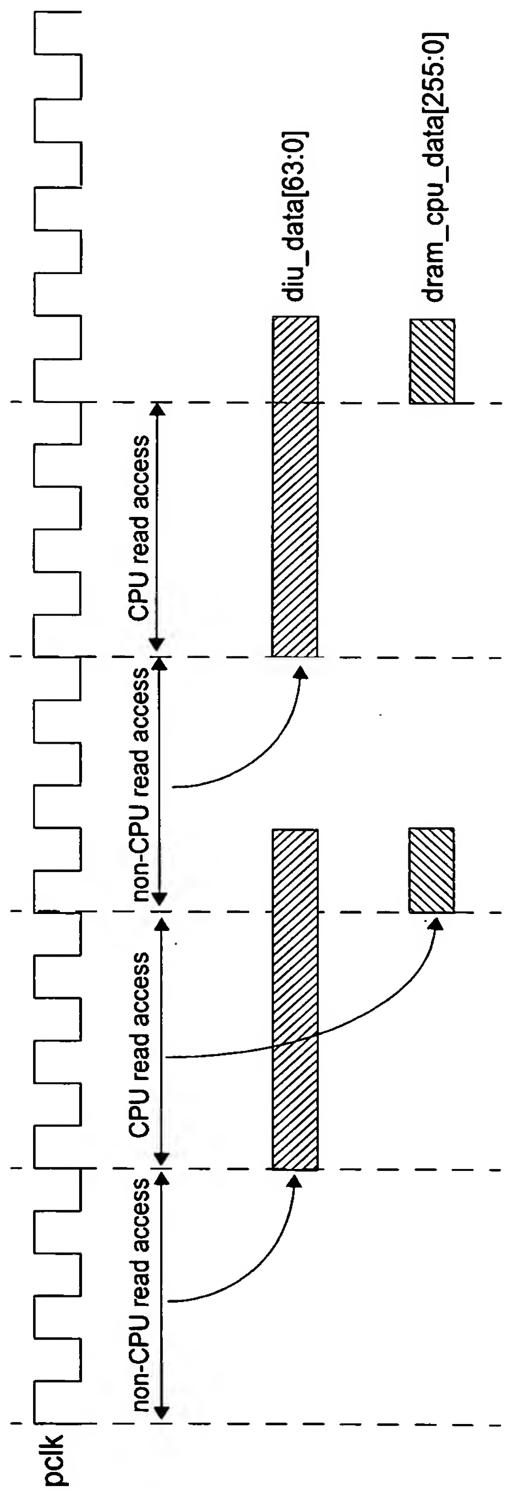

pclk
non-CPU read access
CPU read access
non-CPU read access
CPU read access
diu_data[63:0]
dram_cpu_data[255:0]

FIG. 87

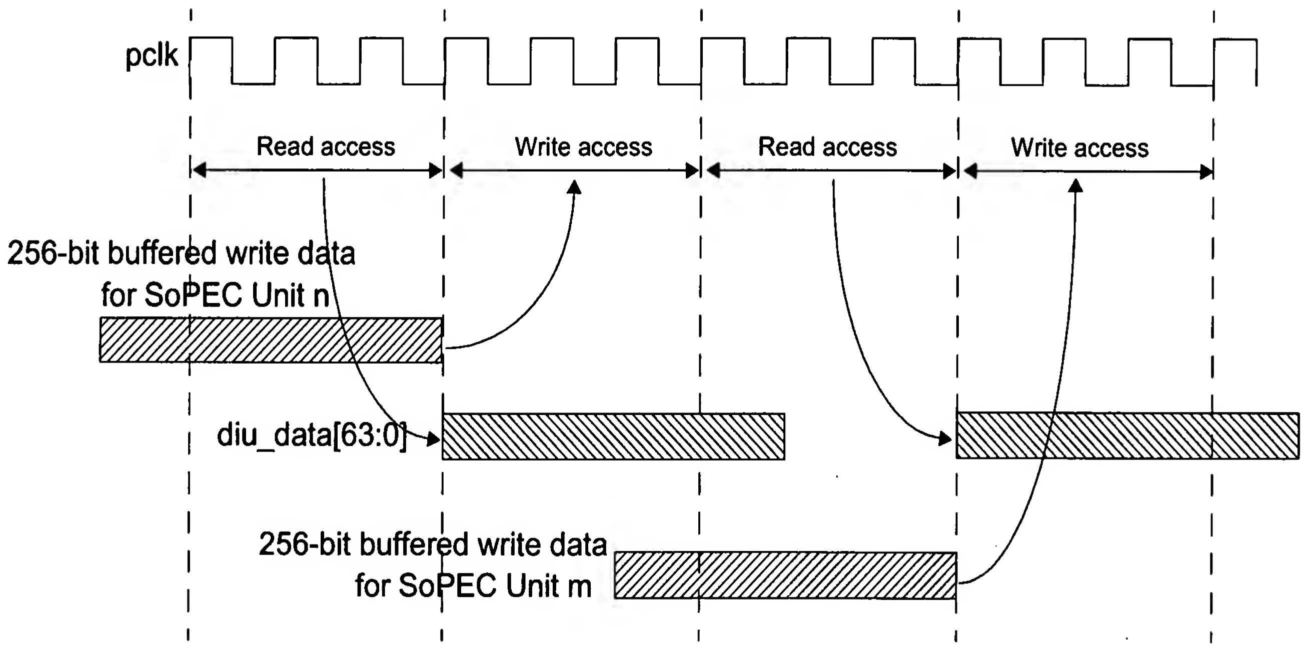

pclk
Read access
Write access
Read access
Write access
256-bit buffered write data
for SoPEC Unit n
diu_data[63:0]
256-bit buffered write data
for SoPEC Unit m

FIG. 88

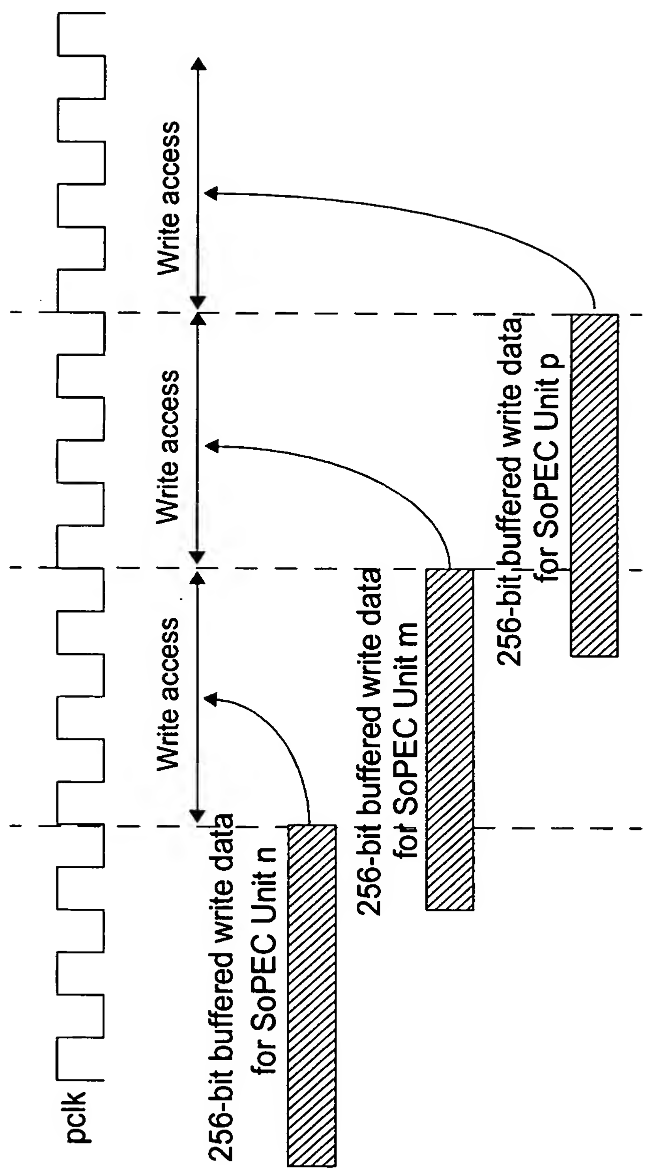

pclk
Write access
Write access
Write access
256-bit buffered write data for SoPEC Unit n
256-bit buffered write data for SoPEC Unit m
256-bit buffered write data for SoPEC Unit p

FIG. 89

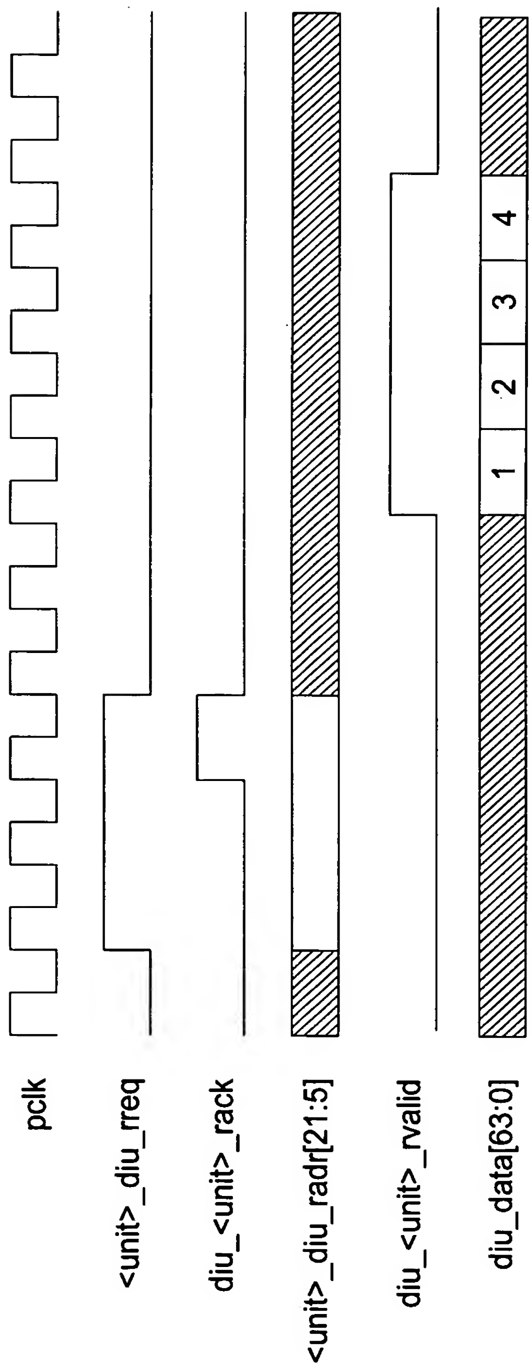

pclk
<unit>_diu_rreq
diu_<unit>_rack
<unit>_diu_radr[21:5]
diu_<unit>_rvalid
diu_data[63:0]
1
2
3
4

FIG. 90

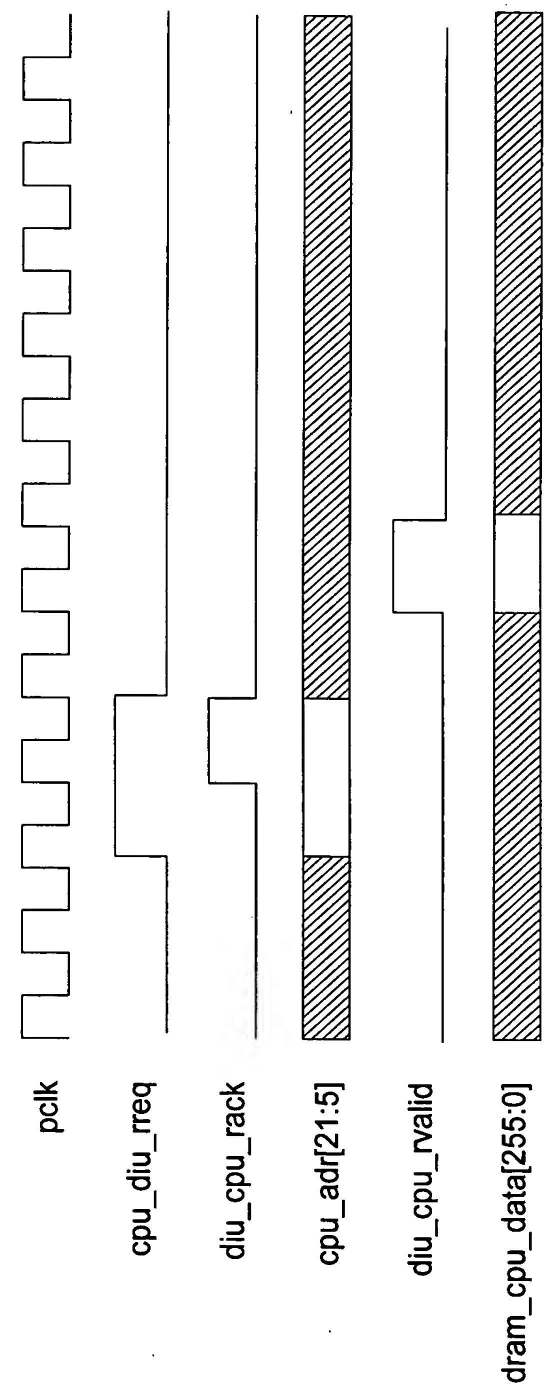

pclk
cpu_diu_rreq
diu_cpu_rack
cpu_adr[21:5]
diu_cpu_rvalid
dram_cpu_data[255:0]

FIG. 91

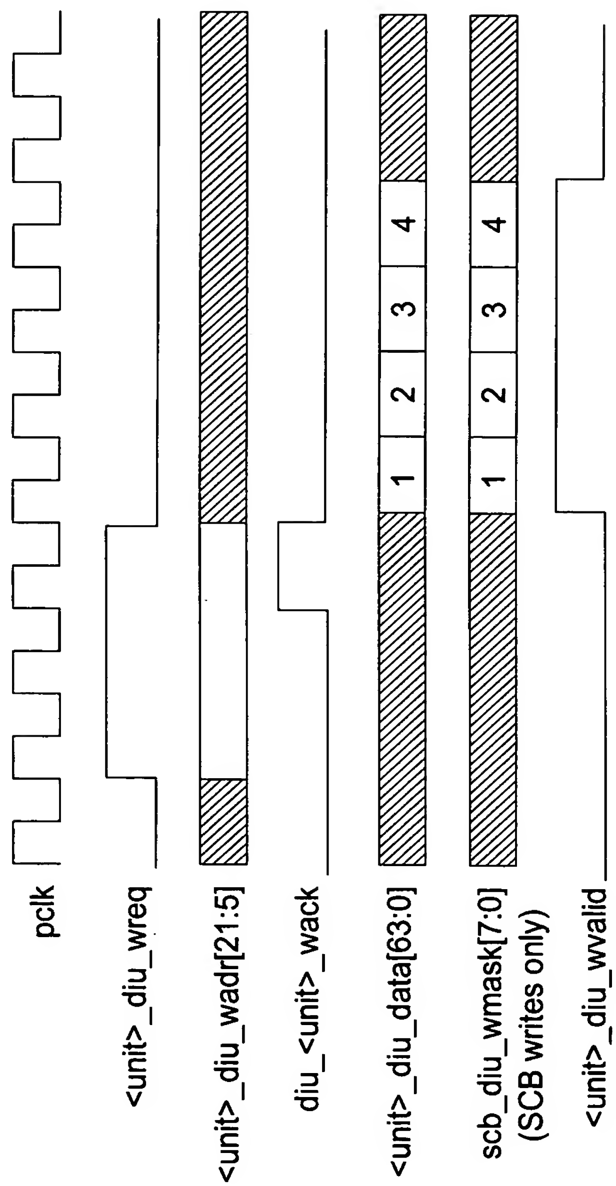

pclk
<unit>_diu_wreq
<unit>_diu_wadr[21:5]
diu_<unit>_wack
<unit>_diu_data[63:0]
scb_diu_wmask[7:0]
(SCB writes only)
<unit>_diu_wvalid
1
2
3
4
1
2
3
4

FIG. 92

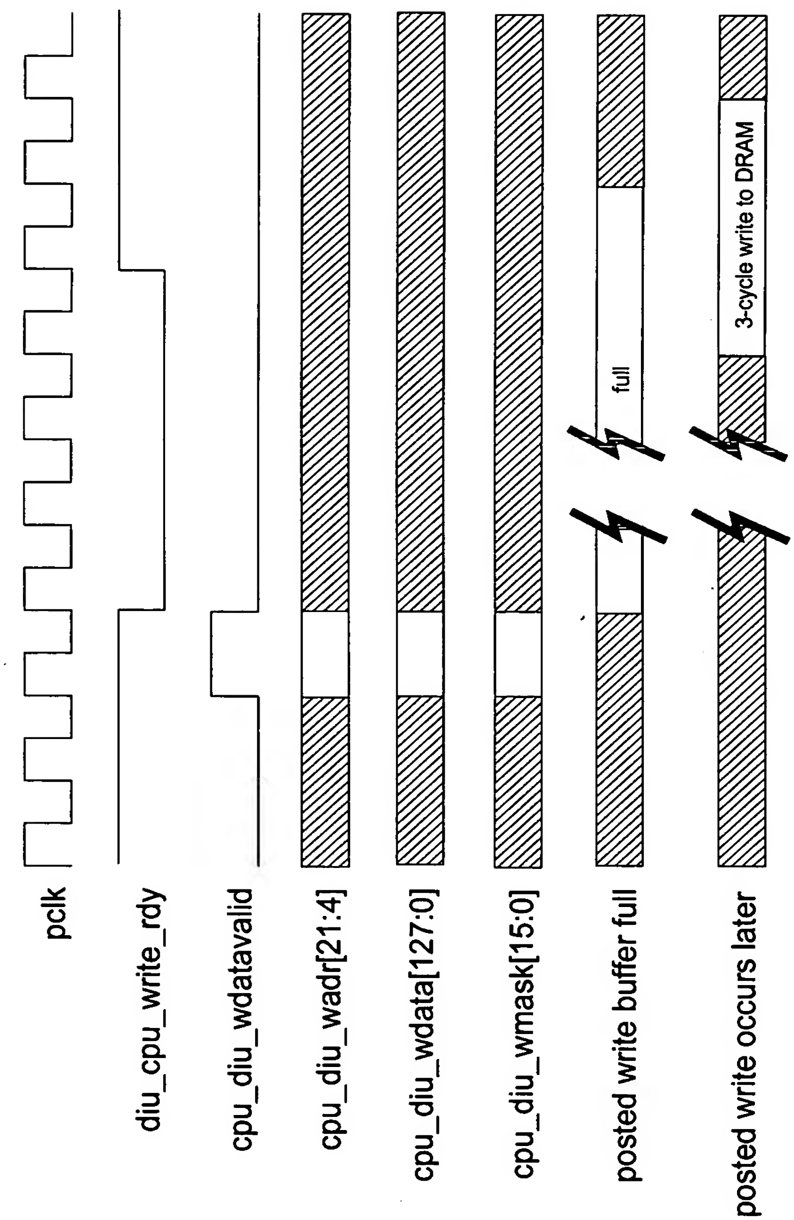

pclk
diu_cpu_write_rdy
cpu_diu_wdatavalid
cpu_diu_wadr[21:4]
cpu_diu_wdata[127:0]
cpu_diu_wmask[15:0]
posted write buffer full
posted write occurs later
full
3-cycle write to DRAM

FIG. 93

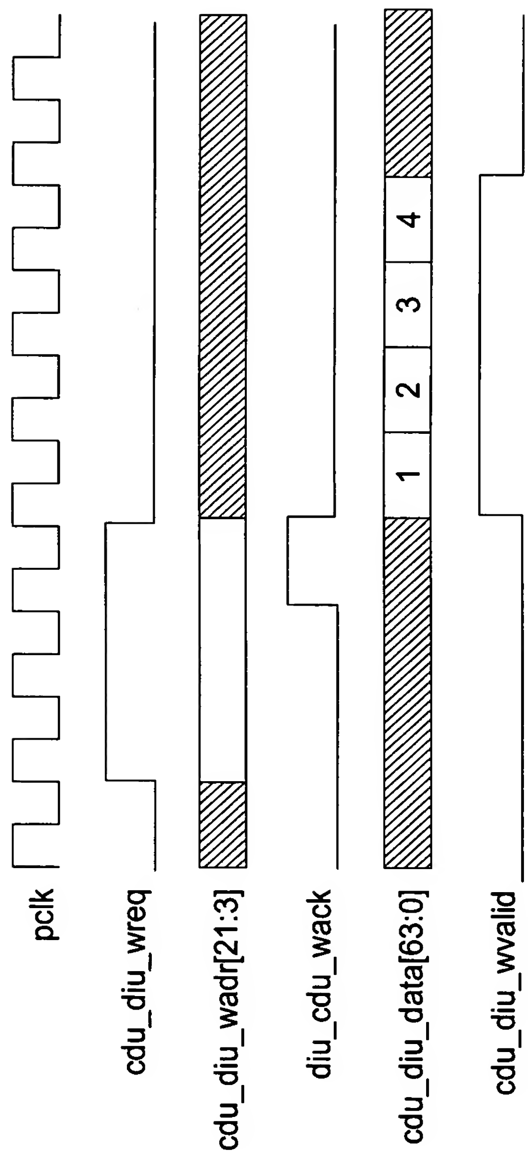

pclk
cdu_diu_wreq
cdu_diu_wadr[21:3]
diu_cdu_wack
cdu_diu_data[63:0]
cdu_diu_wvalid
1
2
3
4

FIG. 94

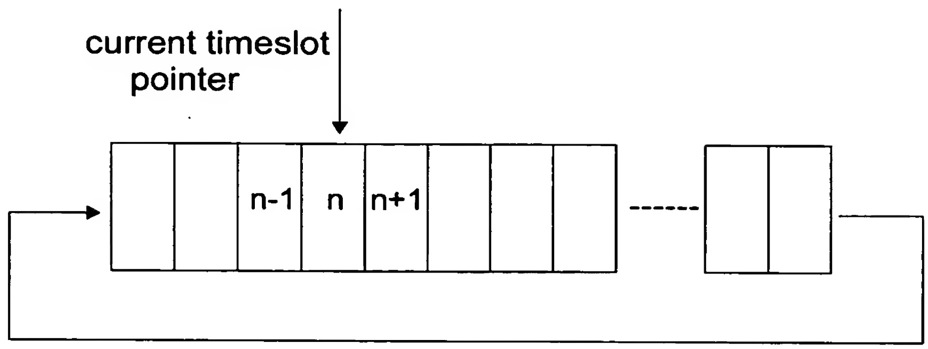


FIG. 95

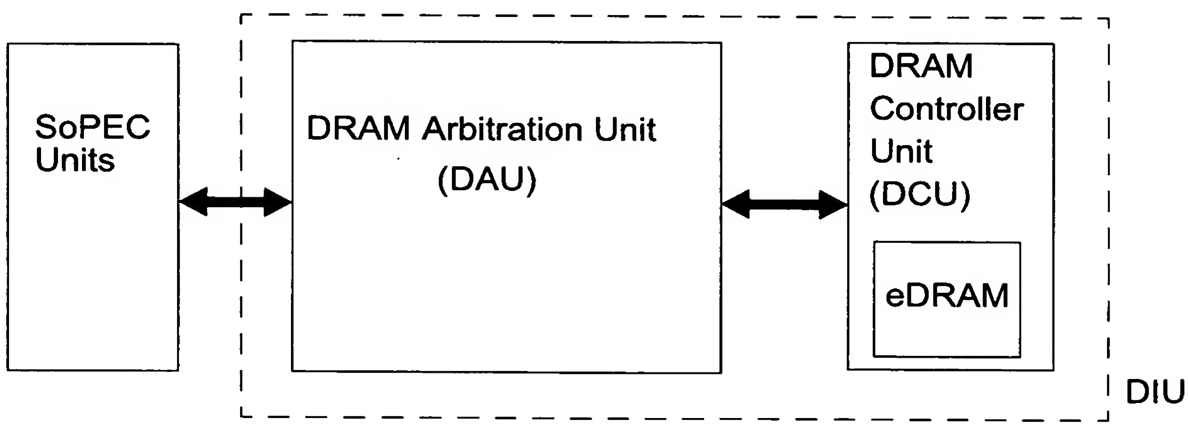


FIG. 100

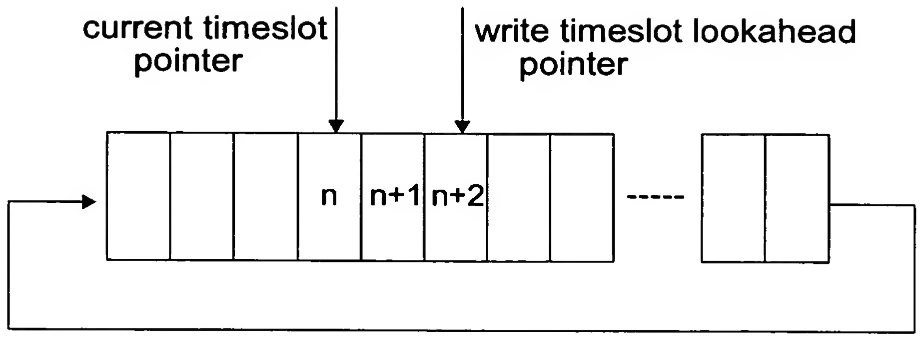

current timeslot pointer
write timeslot lookahead pointer
n
n+1
n+2

FIG. 96

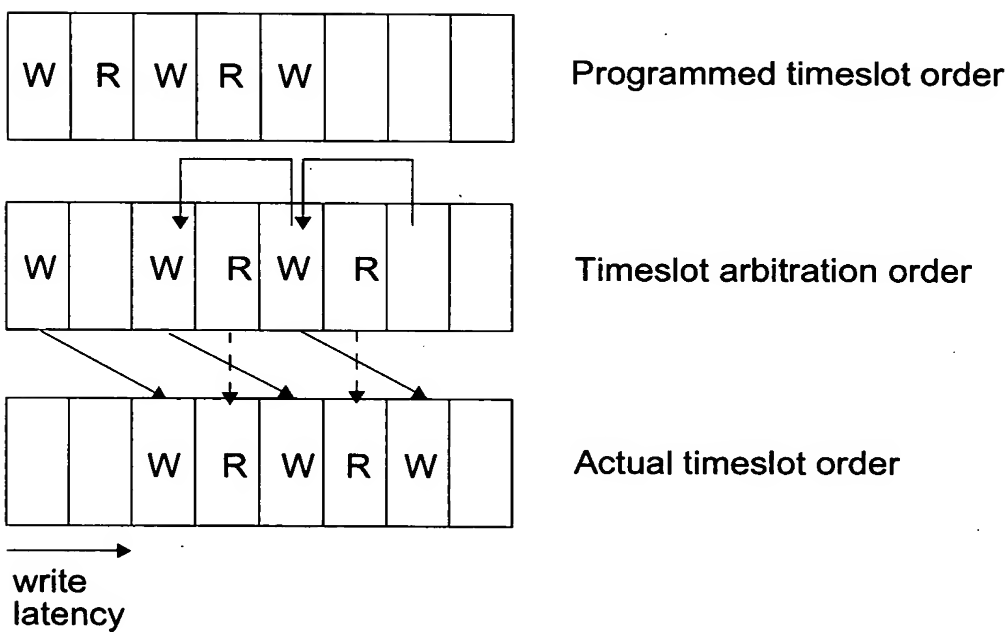

W
R
W
R
W
Programmed timeslot order
W
W
R
W
R
Timeslot arbitration order
W
R
W
R
W
Actual timeslot order
write latency

FIG. 97

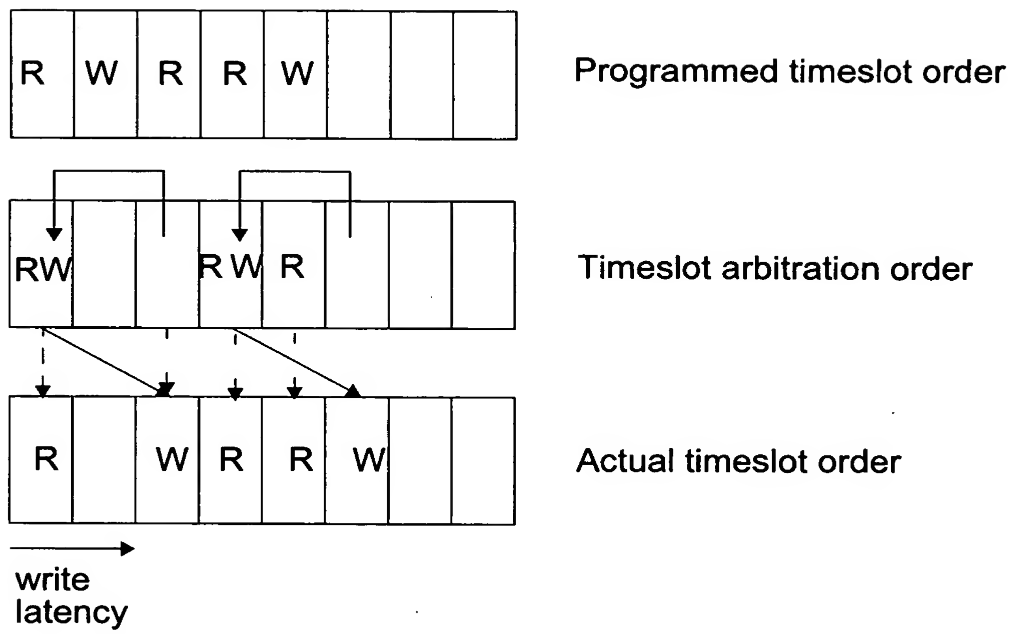

R
W
R
R
W
Programmed timeslot order
RW
R W
R
Timeslot arbitration order
R
W
R
R
W
Actual timeslot order
write
latency

FIG. 98

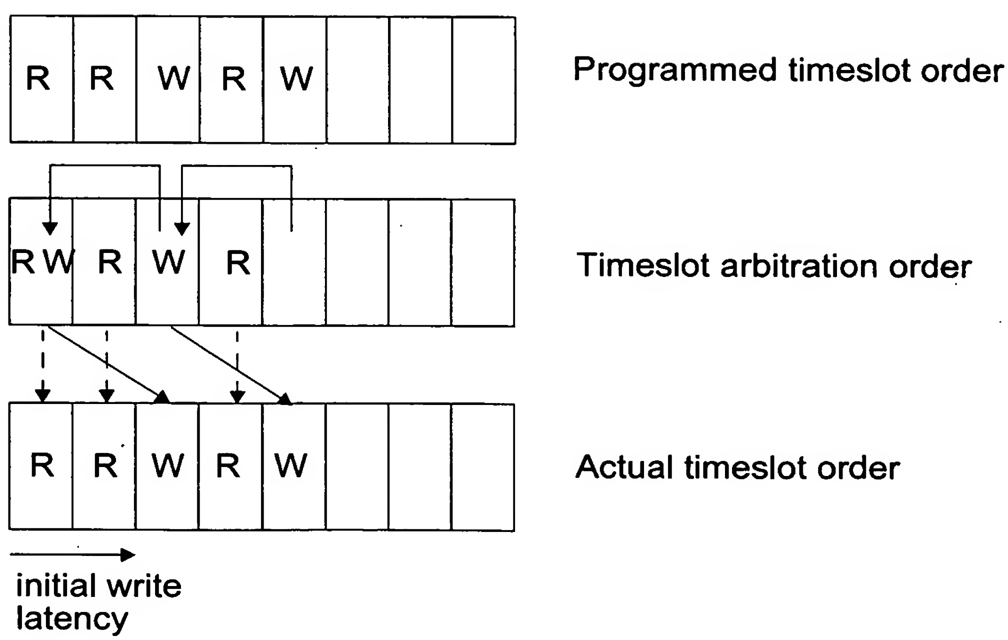

R
R
W
R
W
Programmed timeslot order
R W
R
W
R
Timeslot arbitration order
R
R
W
R
W
Actual timeslot order
initial write
latency

FIG. 99

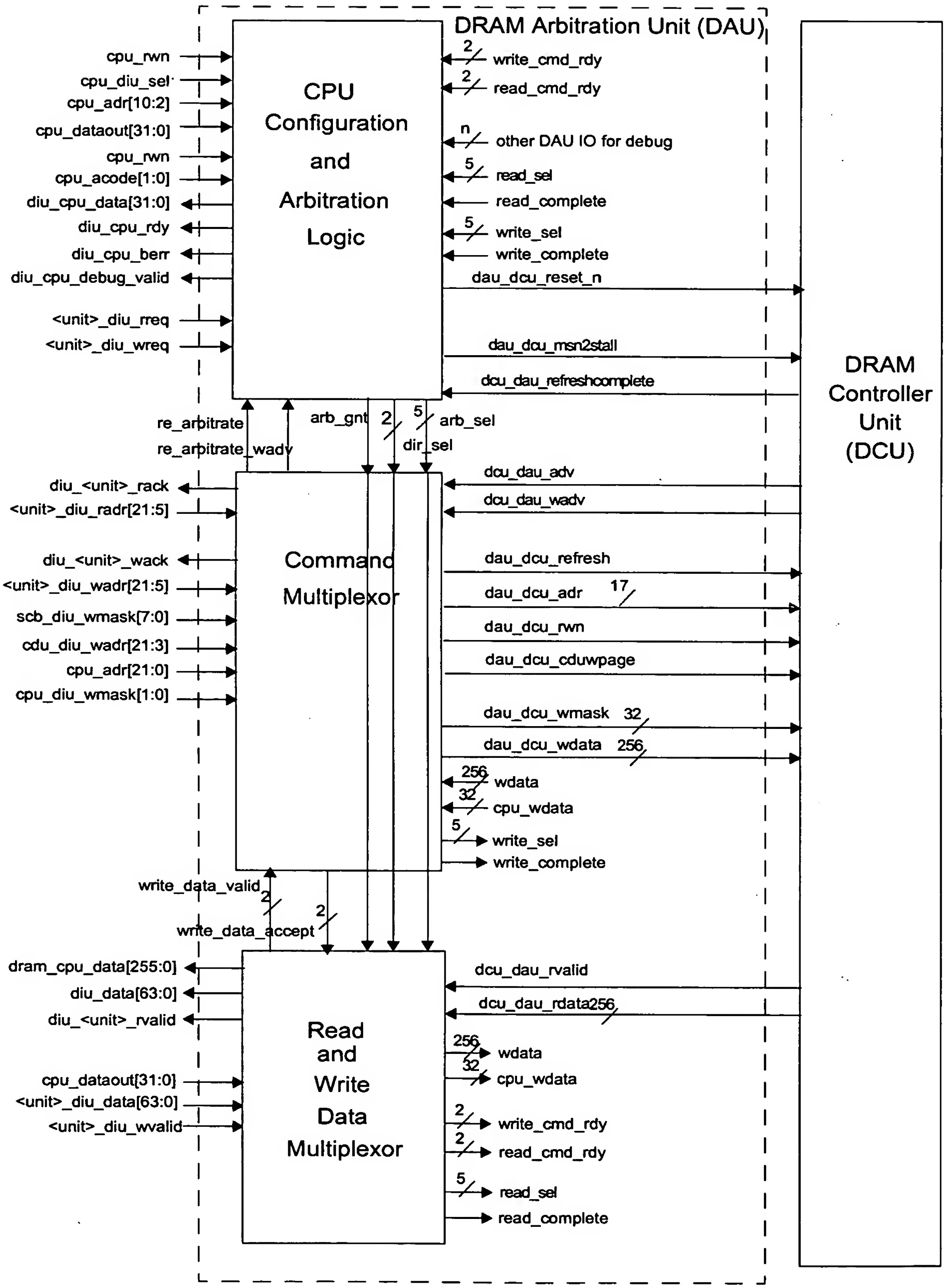
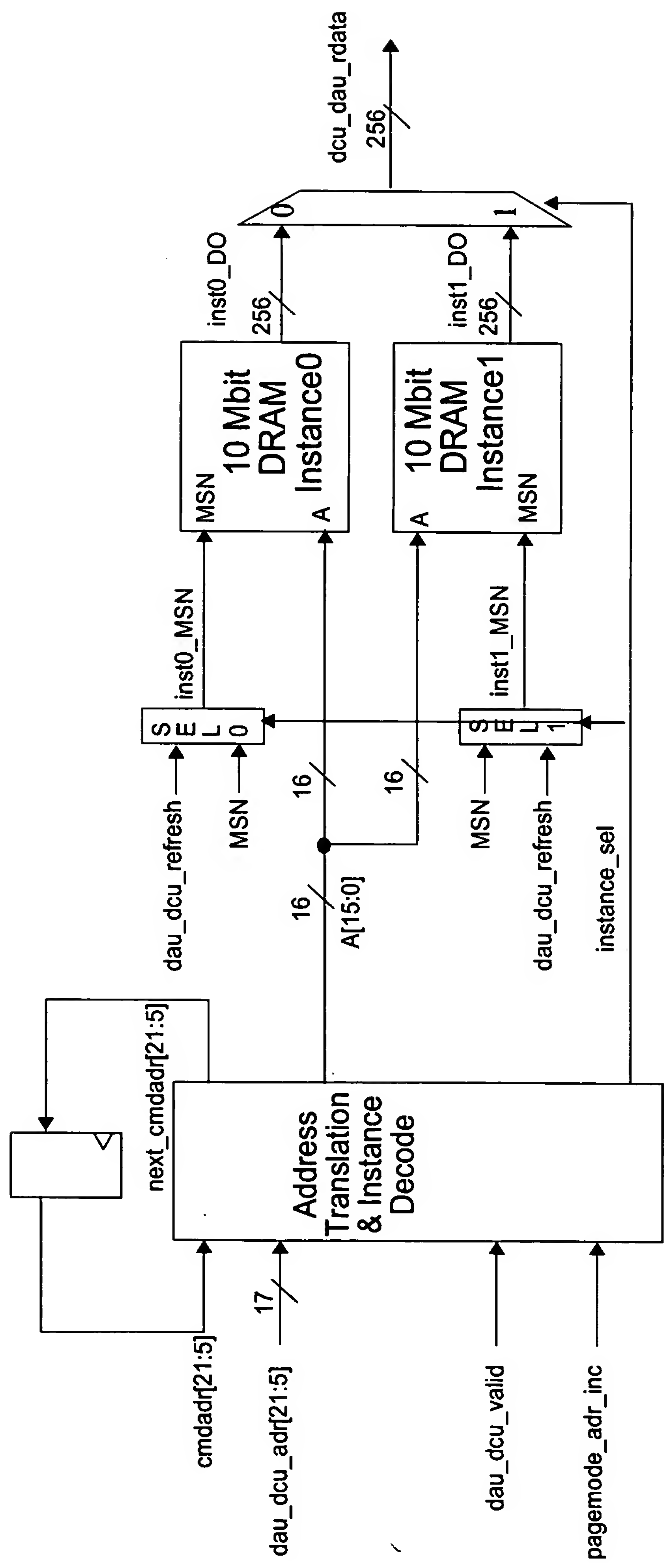

DRAM Arbitration Unit (DAU)
CPU Configuration and Arbitration Logic
cpu_rwn
cpu_diu_sel
cpu_adr[10:2]
cpu_dataout[31:0]
cpu_rwn
cpu_acode[1:0]
diu_cpu_data[31:0]
diu_cpu_rdy
diu_cpu_berr
diu_cpu_debug_valid
<unit>_diu_rreq
<unit>_diu_wreq
2 write_cmd_rdy
2 read_cmd_rdy
n other DAU IO for debug
5 read_sel
read_complete
5 write_sel
write_complete
dau_dcu_reset_n
dau_dcu_msn2stall
dcu_dau_refreshcomplete
re_arbitrate
re_arbitrate_wadv
arb_gnt
2
5
arb_sel
dir_sel
Command Multiplexor
diu_<unit>_rack
<unit>_diu_radr[21:5]
diu_<unit>_wack
<unit>_diu_wadr[21:5]
scb_diu_wmask[7:0]
cdu_diu_wadr[21:3]
cpu_adr[21:0]
cpu_diu_wmask[1:0]
dcu_dau_adv
dcu_dau_wadv
dau_dcu_refresh
dau_dcu_adr 17
dau_dcu_rwn
dau_dcu_cduwpage
dau_dcu_wmask 32
dau_dcu_wdata 256
256 wdata
32 cpu_wdata
5 write_sel
write_complete
DRAM Controller Unit (DCU)
write_data_valid
2
2
write_data_accept
Read and Write Data Multiplexor
dram_cpu_data[255:0]
diu_data[63:0]
diu_<unit>_rvalid
cpu_dataout[31:0]
<unit>_diu_data[63:0]
<unit>_diu_wvalid
dcu_dau_rvalid
dcu_dau_rdata 256
256 wdata
32 cpu_wdata
2 write_cmd_rdy
2 read_cmd_rdy
5 read_sel
read_complete

FIG. 101



dcu_dau_rdata
256
0
1
inst0_DO
256
inst1_DO
256
10 Mbit
DRAM
Instance0
MSN
A
10 Mbit
DRAM
Instance1
A
MSN
inst0_MSN
inst1_MSN
SEL0
SEL1
dau_dcu_refresh
MSN
16
16
16
A[15:0]
MSN
dau_dcu_refresh
instance_sel
next_cmdadr[21:5]
Address
Translation
& Instance
Decode
cmdadr[21:5]
17
dau_dcu_adr[21:5]
dau_dcu_valid
pagemode_adr_inc

FIG. 102

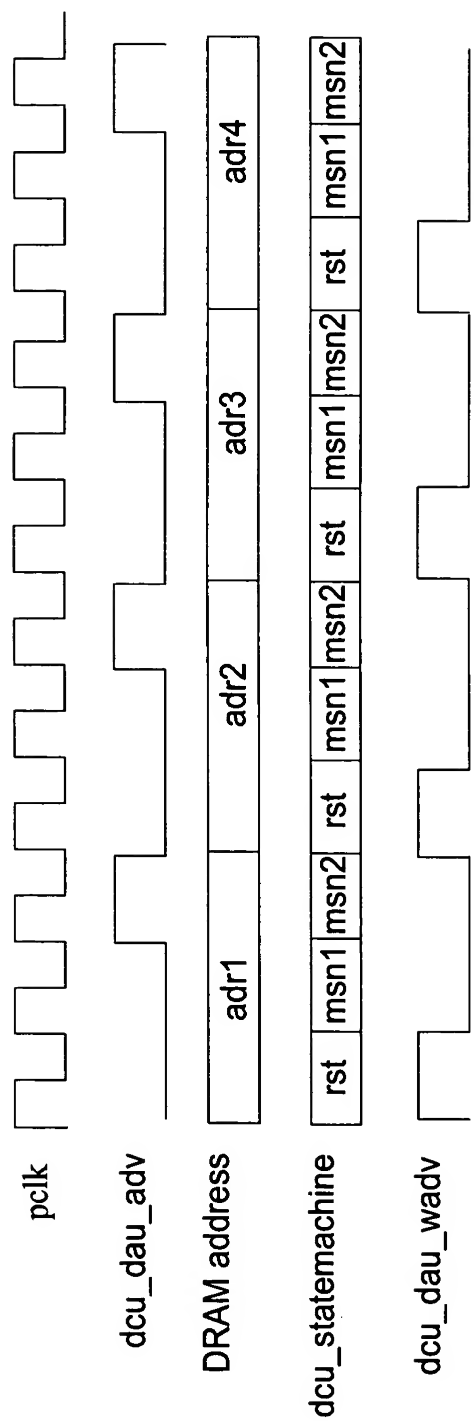

pclk
dcu_dau_adv
DRAM address
adr1
adr2
adr3
adr4
dcu_statemachine
rst
msn1
msn2
dcu_dau_wadv

FIG. 103

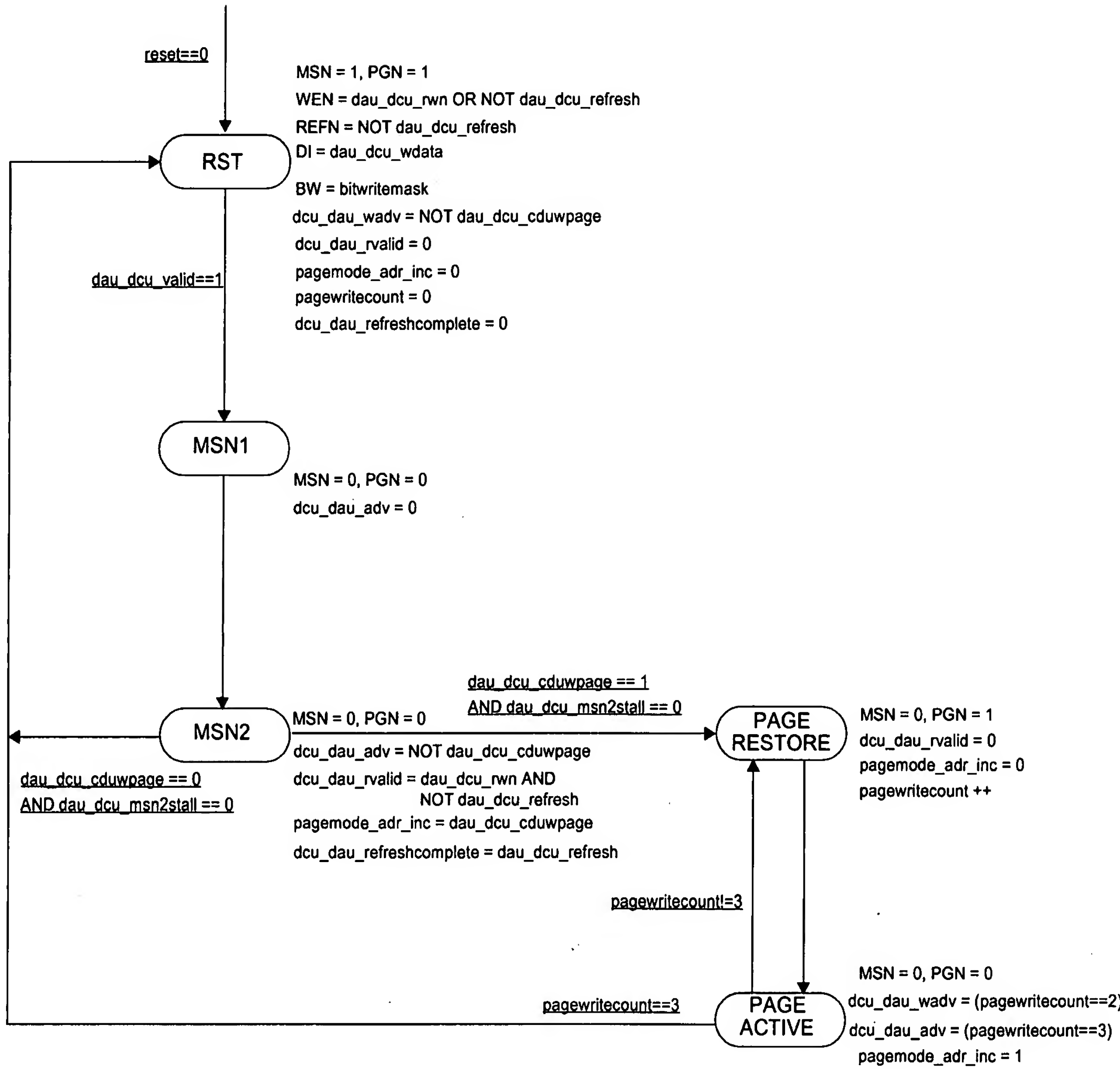

reset==0
RST
MSN = 1, PGN = 1
WEN = dau_dcu_rwn OR NOT dau_dcu_refresh
REFN = NOT dau_dcu_refresh
DI = dau_dcu_wdata
BW = bitwritemask
dcu_dau_wadv = NOT dau_dcu_cduwpage
dcu_dau_rvalid = 0
pagemode_adr_inc = 0
pagewritecount = 0
dcu_dau_refreshcomplete = 0
dau_dcu_valid==1
MSN1
MSN = 0, PGN = 0
dcu_dau_adv = 0
MSN2
MSN = 0, PGN = 0
dcu_dau_adv = NOT dau_dcu_cduwpage
dcu_dau_rvalid = dau_dcu_rwn AND NOT dau_dcu_refresh
pagemode_adr_inc = dau_dcu_cduwpage
dcu_dau_refreshcomplete = dau_dcu_refresh
dau_dcu_cduwpage == 0 AND dau_dcu_msn2stall == 0
dau_dcu_cduwpage == 1 AND dau_dcu_msn2stall == 0
PAGE RESTORE
MSN = 0, PGN = 1
dcu_dau_rvalid = 0
pagemode_adr_inc = 0
pagewritecount ++
pagewritecount!=3
pagewritecount==3
PAGE ACTIVE
MSN = 0, PGN = 0
dcu_dau_wadv = (pagewritecount==2)
dcu_dau_adv = (pagewritecount==3)
pagemode_adr_inc = 1

FIG. 104

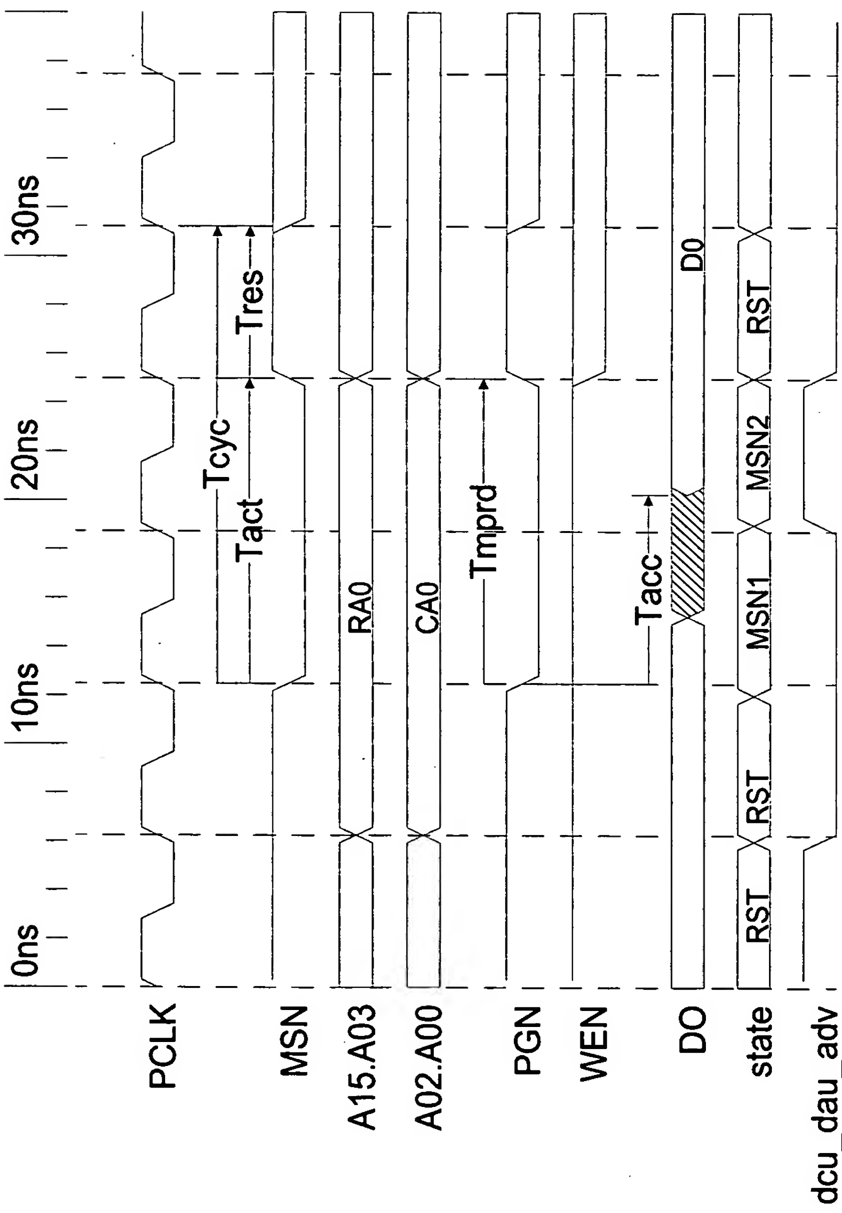

0ns
10ns
20ns
30ns
PCLK
MSN
A15.A03
A02.A00
PGN
WEN
DO
state
dcu_dau_adv
Tcyc
Tact
Tres
Tmprd
Tacc
RA0
CA0
D0
RST
RST
MSN1
MSN2
RST

FIG. 105

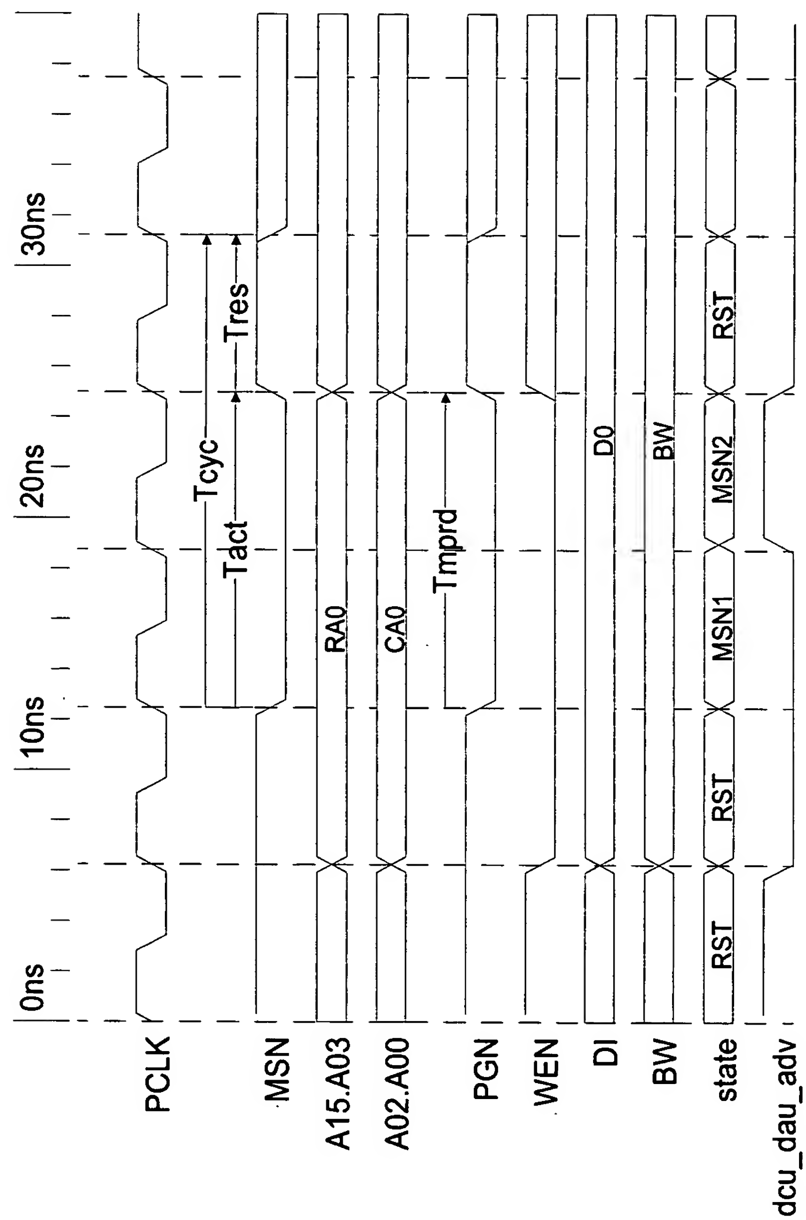

0ns
10ns
20ns
30ns
PCLK
MSN
A15.A03
A02.A00
PGN
WEN
DI
BW
state
dcu_dau_adv
Tcyc
Tact
Tres
Tmprd
RA0
CA0
D0
BW
RST
RST
MSN1
MSN2
RST

FIG. 106

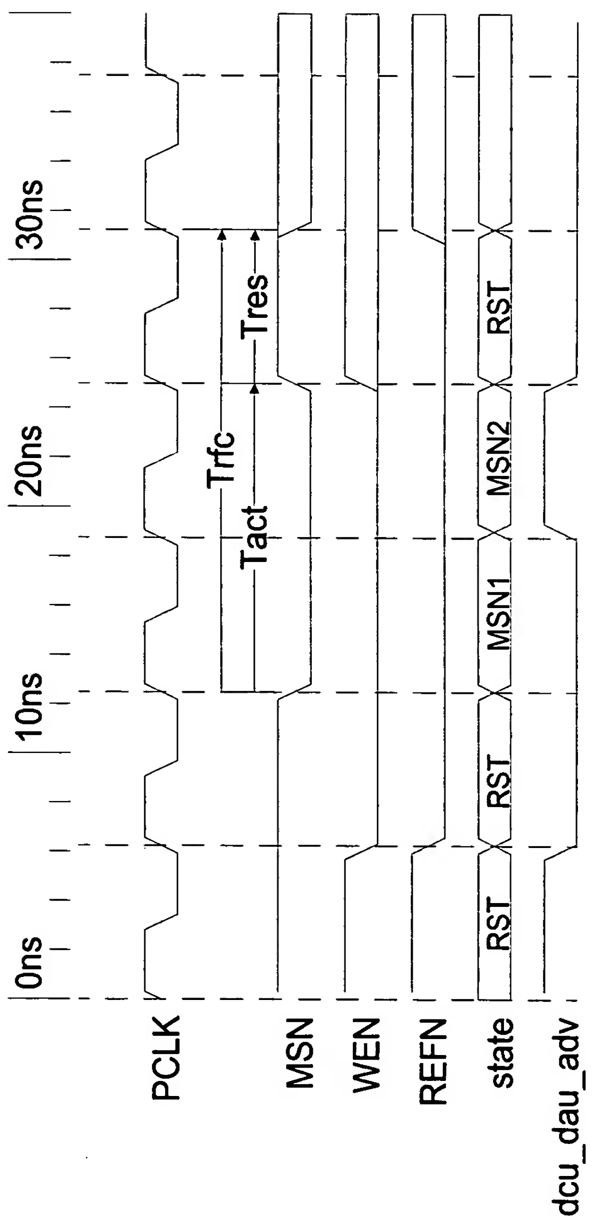

0ns
10ns
20ns
30ns
PCLK
Trfc
Tact
Tres
MSN
WEN
REFN
state
RST
RST
MSN1
MSN2
RST
dcu_dau_adv

FIG. 107

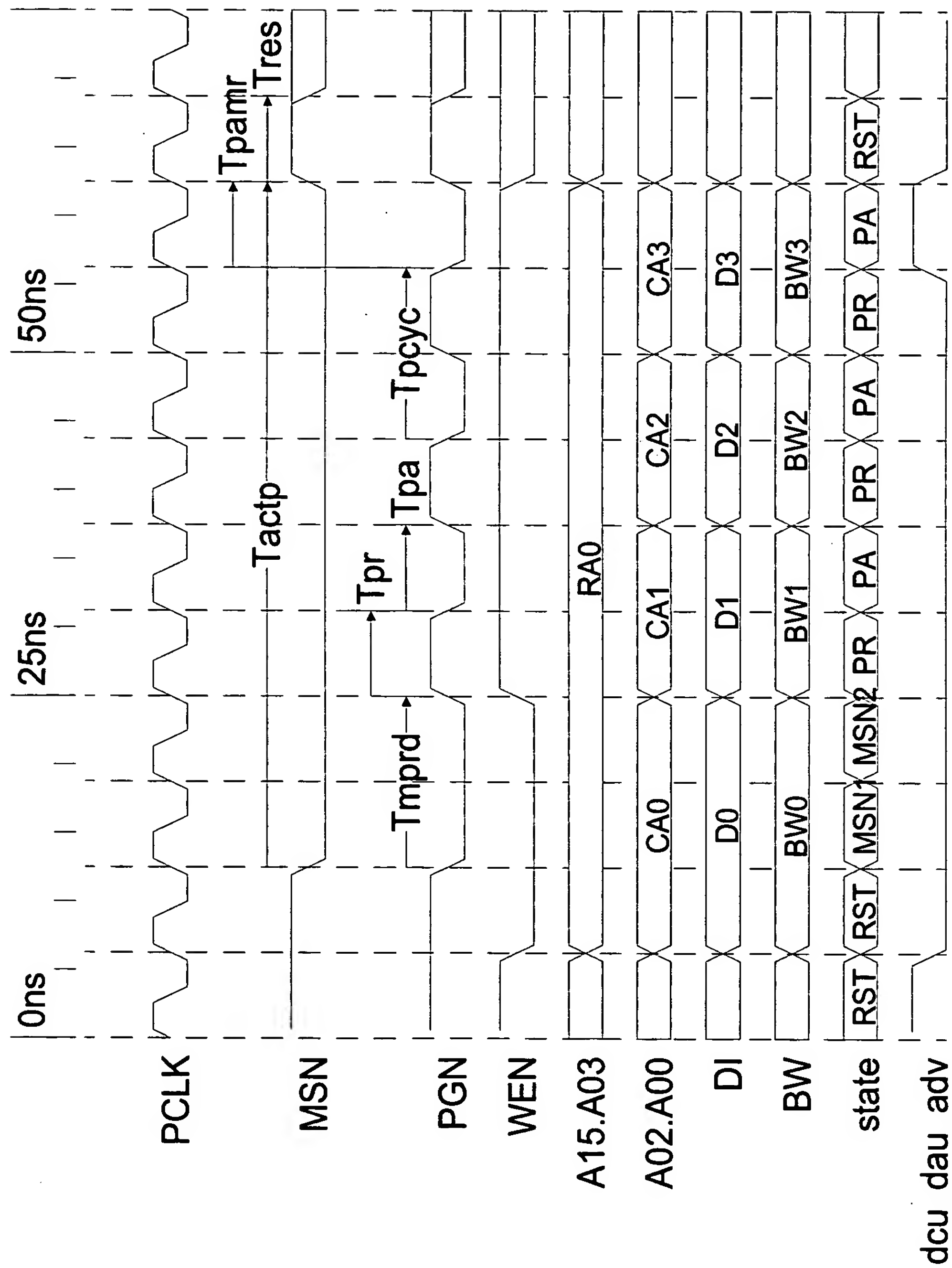

PCLK
MSN
PGN
WEN
A15.A03
A02.A00
DI
BW
state
dcu_dau_adv
0ns
25ns
50ns
Tmprd
Tpr
Tpa
Tpcyc
Tactp
Tpamr
Tres
RA0
CA0
CA1
CA2
CA3
D0
D1
D2
D3
BW0
BW1
BW2
BW3
RST
MSN1
MSN2
PR
PA
PR
PA
PR
PA
RST

FIG. 108

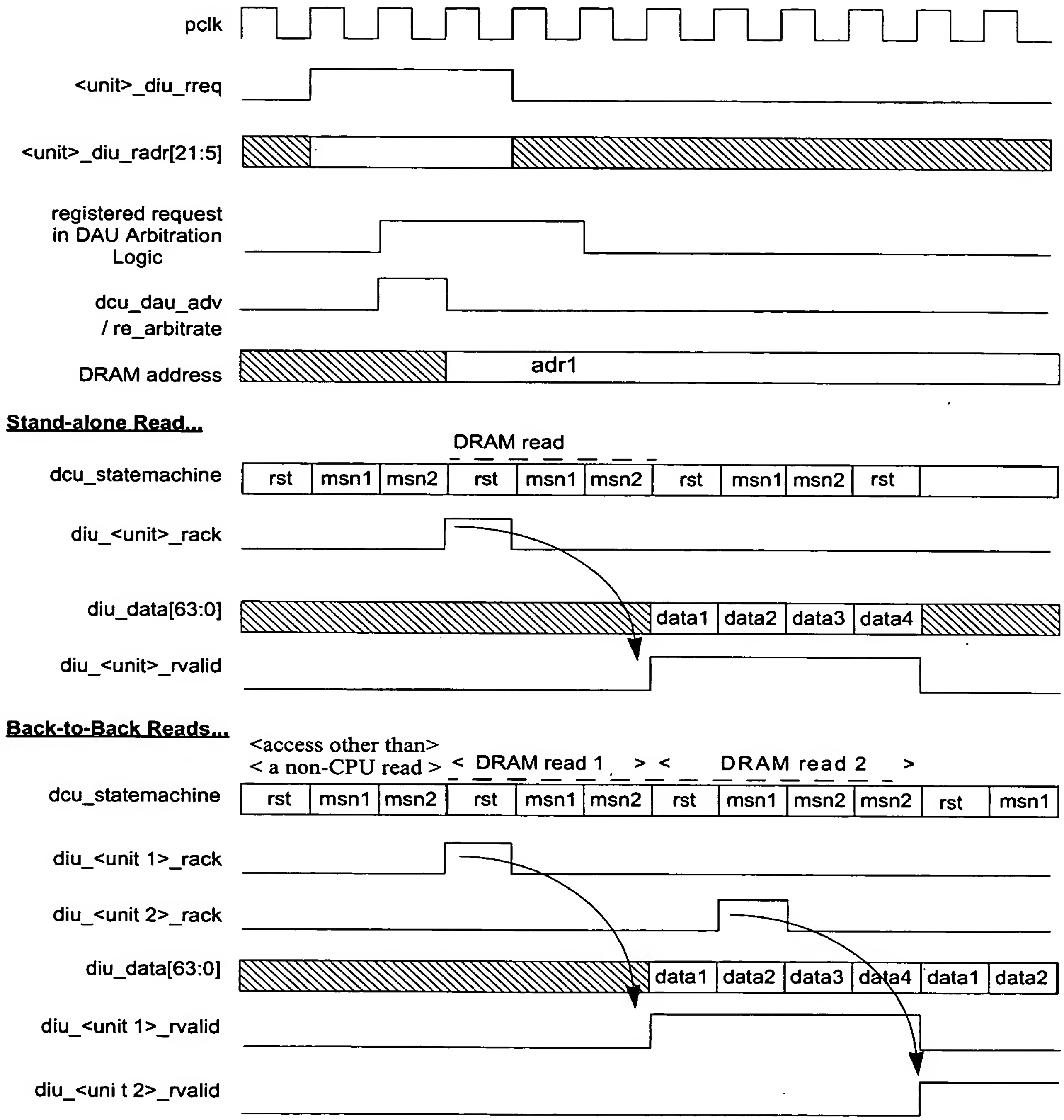

pclk
<unit>_diu_rreq
<unit>_diu_radr[21:5]
registered request in DAU Arbitration Logic
dcu_dau_adv / re_arbitrate
DRAM address
adr1
Stand-alone Read...
DRAM read
dcu_statemachine
rst
msn1
msn2
rst
msn1
msn2
rst
msn1
msn2
rst
diu_<unit>_rack
diu_data[63:0]
data1
data2
data3
data4
diu_<unit>_rvalid
Back-to-Back Reads...
<access other than>
< a non-CPU read >
< DRAM read 1 >
< DRAM read 2 >
dcu_statemachine
rst
msn1
msn2
rst
msn1
msn2
rst
msn1
msn2
msn2
rst
msn1
diu_<unit 1>_rack
diu_<unit 2>_rack
diu_data[63:0]
data1
data2
data3
data4
data1
data2
diu_<unit 1>_rvalid
diu_<uni t 2>_rvalid

FIG. 109

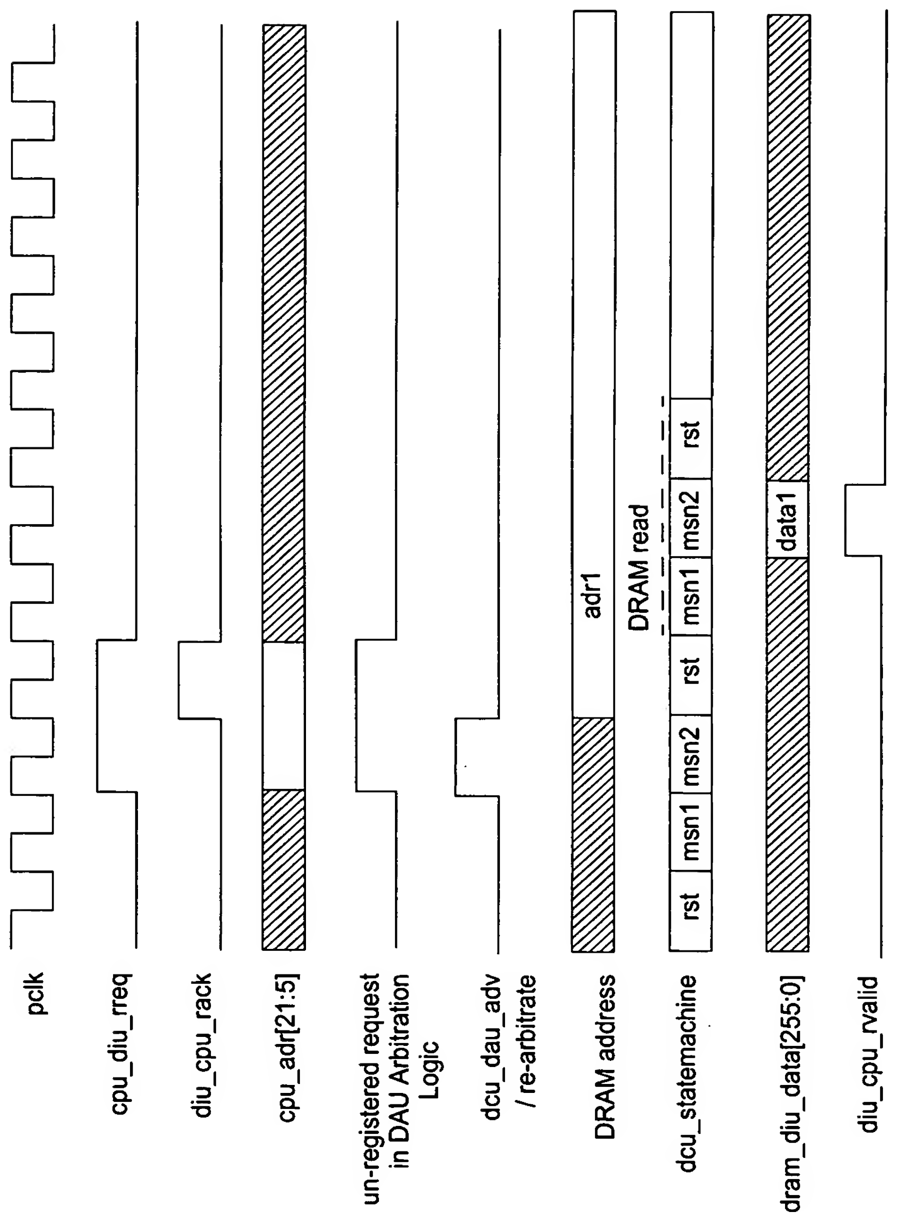

pclk
cpu_diu_rreq
diu_cpu_rack
cpu_adr[21:5]
un-registered request in DAU Arbitration Logic
dcu_dau_adv / re-arbitrate
DRAM address
adr1
dcu_statemachine
rst
msn1
msn2
rst
msn1
msn2
rst
DRAM read
dram_diu_data[255:0]
data1
diu_cpu_rvalid

FIG. 110

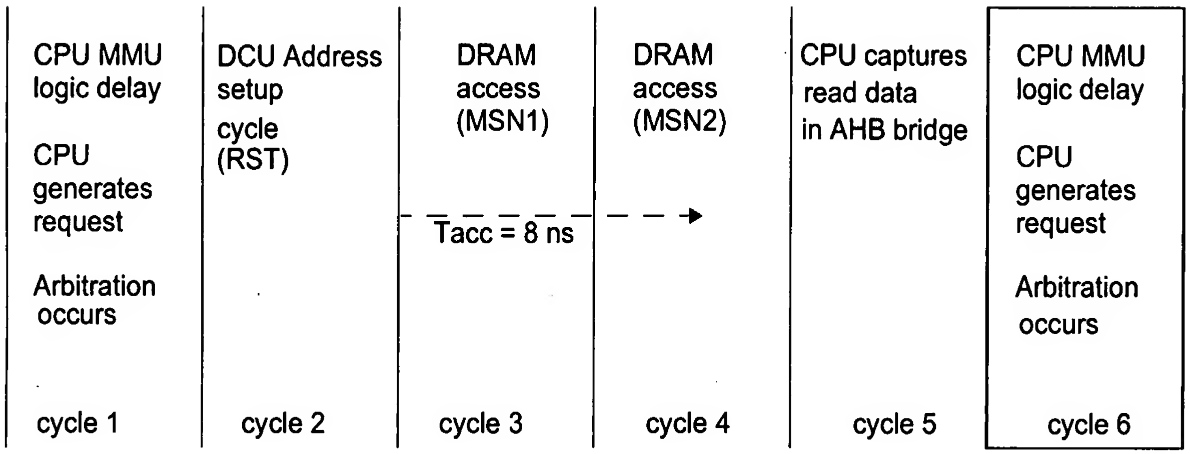

CPU MMU logic delay
CPU generates request
Arbitration occurs
cycle 1
DCU Address setup cycle (RST)
cycle 2
DRAM access (MSN1)
Tacc = 8 ns
cycle 3
DRAM access (MSN2)
cycle 4
CPU captures read data in AHB bridge
cycle 5
CPU MMU logic delay
CPU generates request
Arbitration occurs
cycle 6

FIG. 111

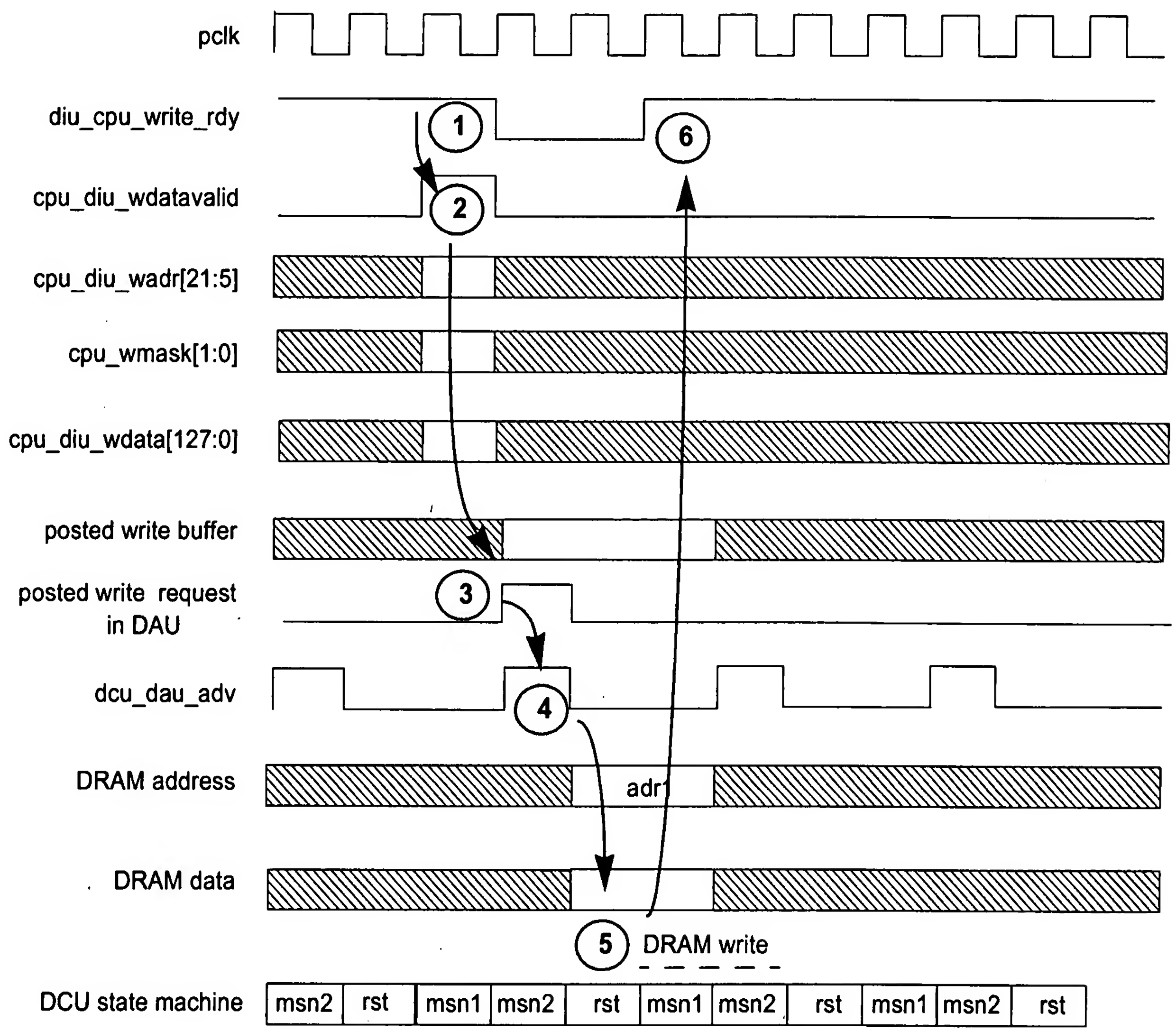

pclk
diu_cpu_write_rdy
cpu_diu_wdatavalid
cpu_diu_wadr[21:5]
cpu_wmask[1:0]
cpu_diu_wdata[127:0]
posted write buffer
posted write request in DAU
dcu_dau_adv
DRAM address
adr1
DRAM data
DRAM write
DCU state machine
msn2
rst
msn1
msn2
rst
msn1
msn2
rst
msn1
msn2
rst
1
2
3
4
5
6

FIG. 112

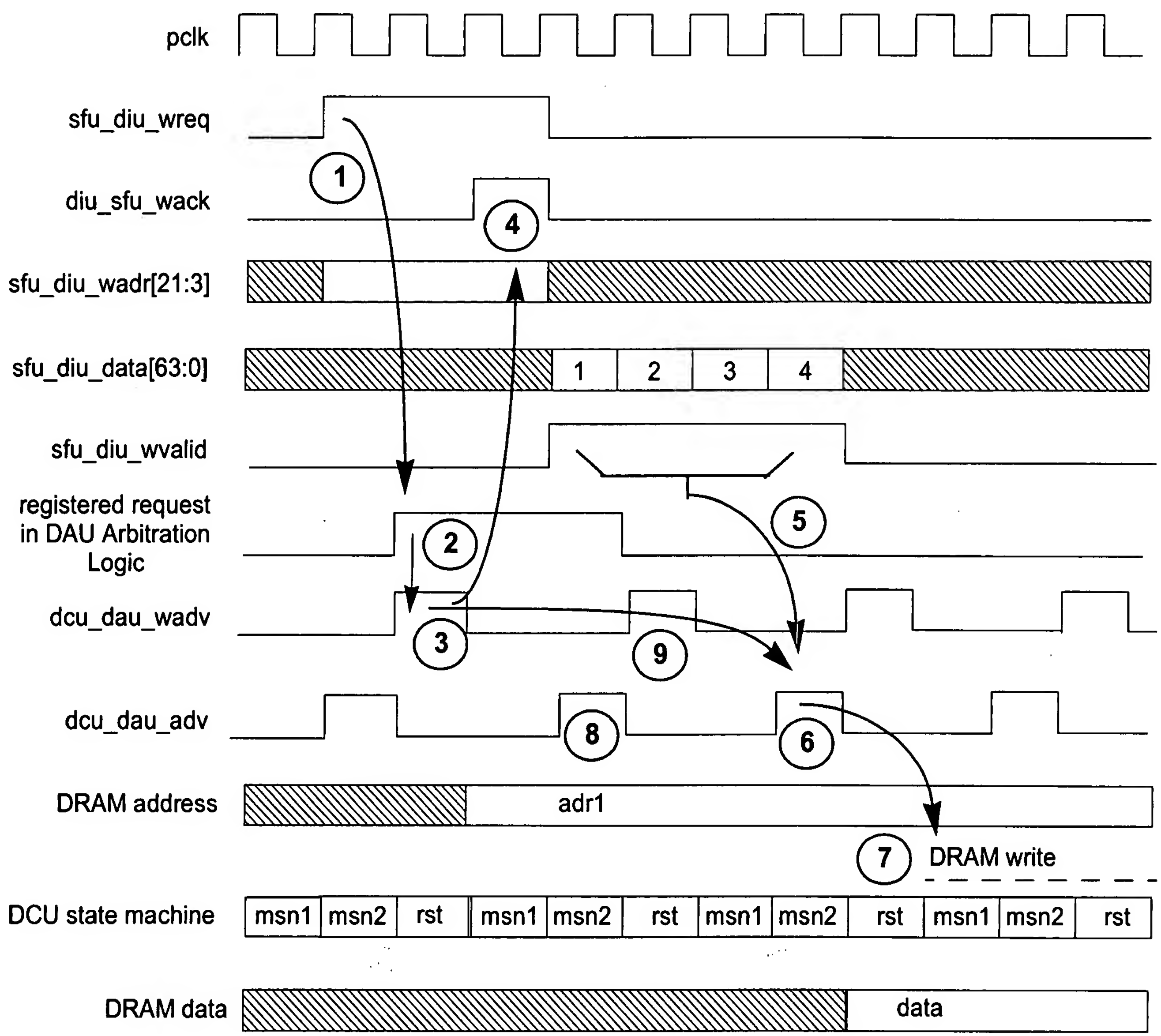

pclk
sfu_diu_wreq
diu_sfu_wack
sfu_diu_wadr[21:3]
sfu_diu_data[63:0]
1
2
3
4
sfu_diu_wvalid
registered request in DAU Arbitration Logic
dcu_dau_wadv
dcu_dau_adv
DRAM address
adr1
DRAM write
DCU state machine
msn1
msn2
rst
msn1
msn2
rst
msn1
msn2
rst
msn1
msn2
rst
DRAM data
data
1
2
3
4
5
6
7
8
9

FIG. 113

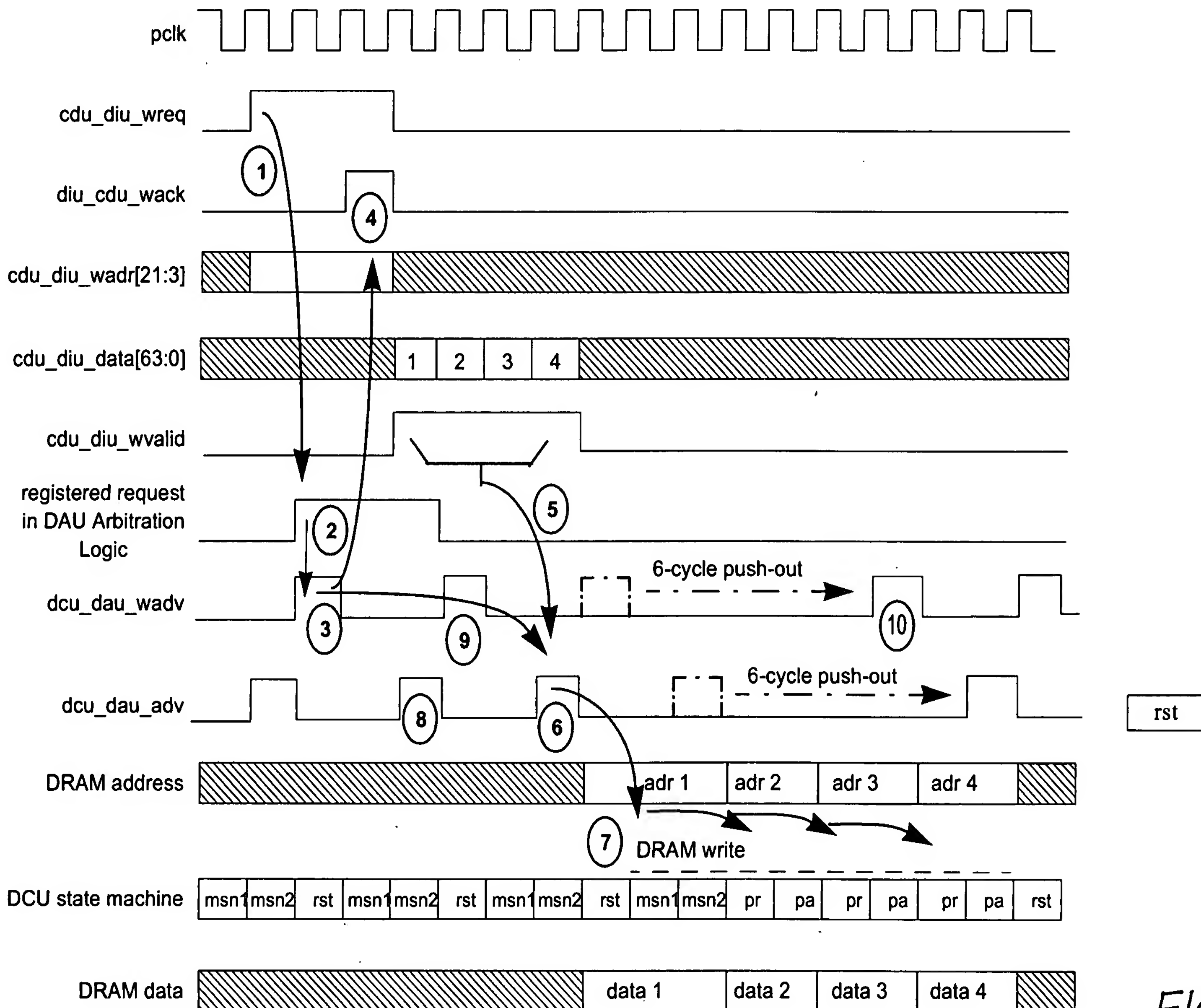

pclk
cdu_diu_wreq
diu_cdu_wack
cdu_diu_wadr[21:3]
cdu_diu_data[63:0]
1
2
3
4
cdu_diu_wvalid
registered request in DAU Arbitration Logic
dcu_dau_wadv
6-cycle push-out
dcu_dau_adv
6-cycle push-out
rst
DRAM address
adr 1
adr 2
adr 3
adr 4
DRAM write
DCU state machine
msn1
msn2
rst
msn1
msn2
rst
msn1
msn2
rst
msn1
msn2
pr
pa
pr
pa
pr
pa
rst
DRAM data
data 1
data 2
data 3
data 4

FIG. 114

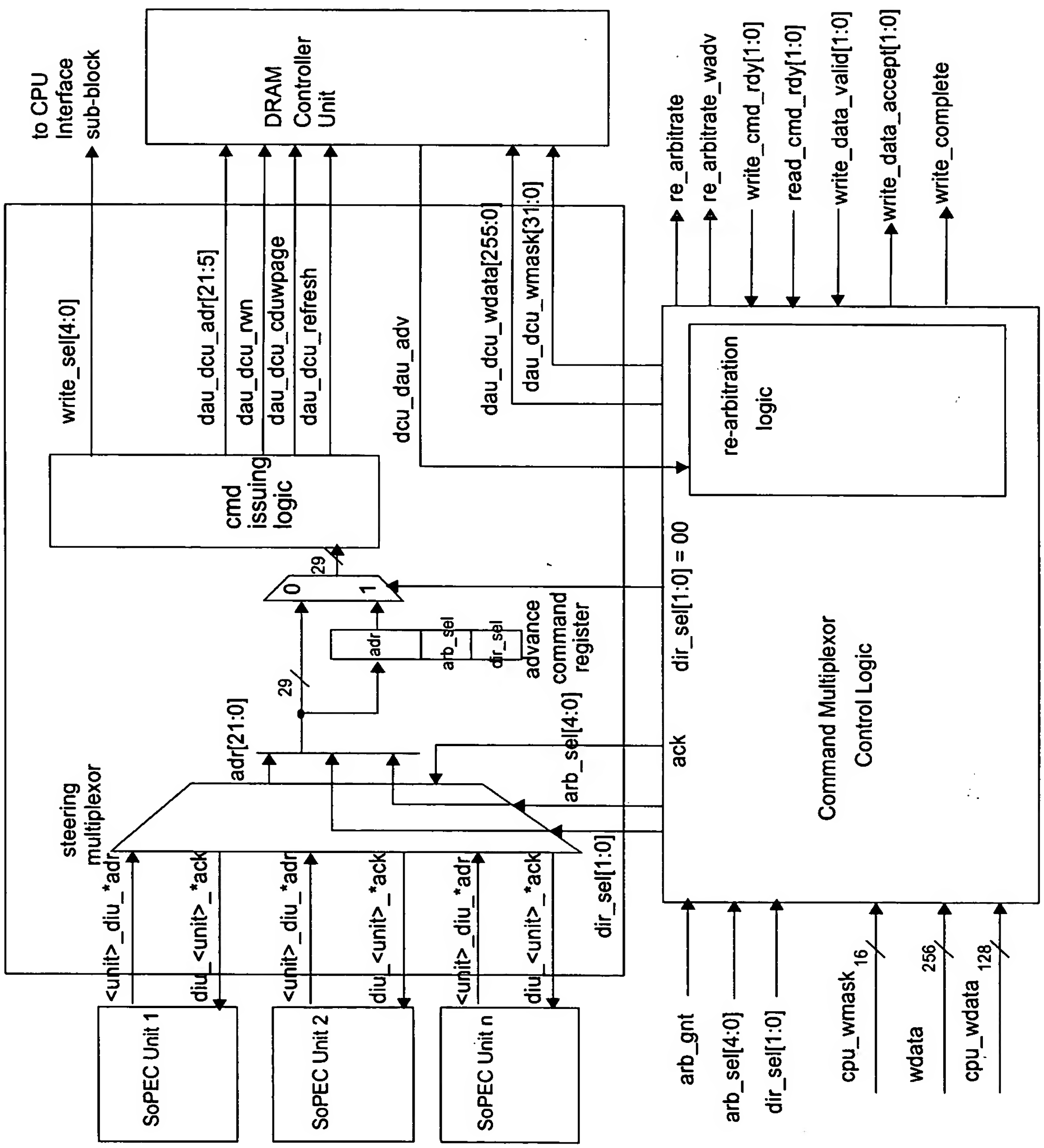

to CPU Interface sub-block
DRAM Controller Unit
write_sel[4:0]
dau_dcu_adr[21:5]
dau_dcu_rwn
dau_dcu_cduwpage
dau_dcu_refresh
dcu_dau_adv
dau_dcu_wdata[255:0]
dau_dcu_wmask[31:0]
re_arbitrate
re_arbitrate_wadv
write_cmd_rdy[1:0]
read_cmd_rdy[1:0]
write_data_valid[1:0]
write_data_accept[1:0]
write_complete
cmd issuing logic
re-arbitration logic
29
0
1
adr
arb_sel
dir_sel
advance command register
dir_sel[1:0] = 00
29
adr[21:0]
steering multiplexor
ack
arb_sel[4:0]
dir_sel[1:0]
Command Multiplexor Control Logic
<unit>_diu_*adr
diu_<unit>_*ack
<unit>_diu_*adr
diu_<unit>_*ack
<unit>_diu_*adr
diu_<unit>_*ack
SoPEC Unit 1
SoPEC Unit 2
SoPEC Unit n
arb_gnt
arb_sel[4:0]
dir_sel[1:0]
cpu_wmask
16
wdata
256
cpu_wdata
128

FIG. 115

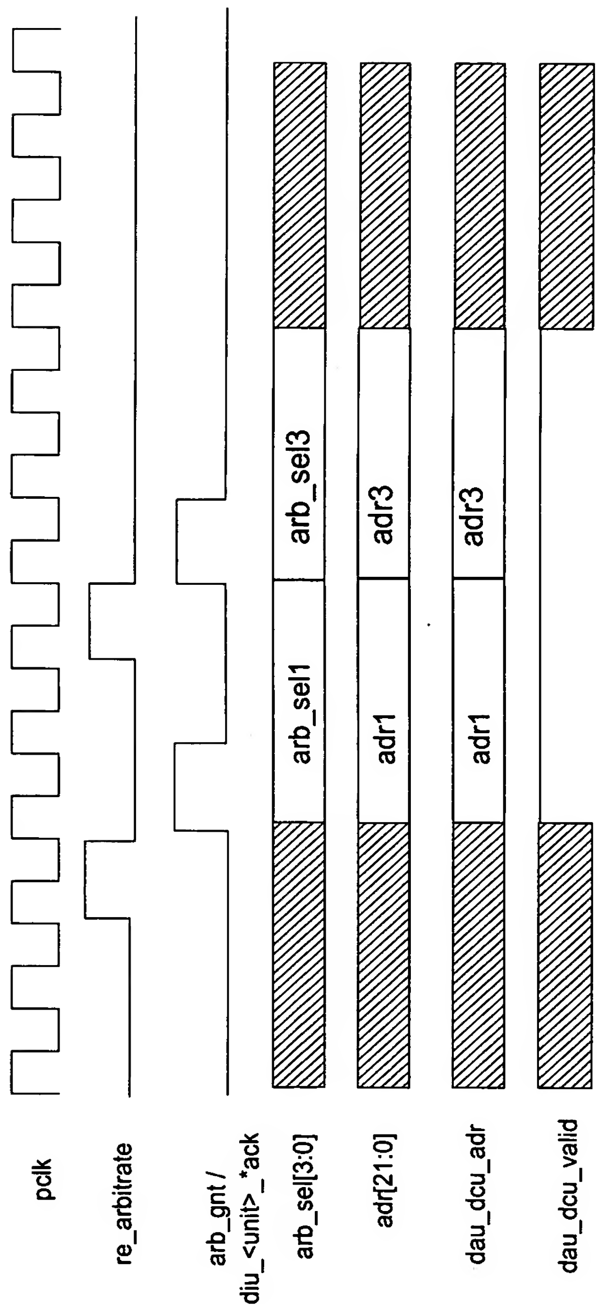

pclk
re_arbitrate
arb_gnt / diu_<unit>_*ack
arb_sel[3:0]
adr[21:0]
dau_dcu_adr
dau_dcu_valid
arb_sel1
arb_sel3
adr1
adr3
adr1
adr3

FIG. 116

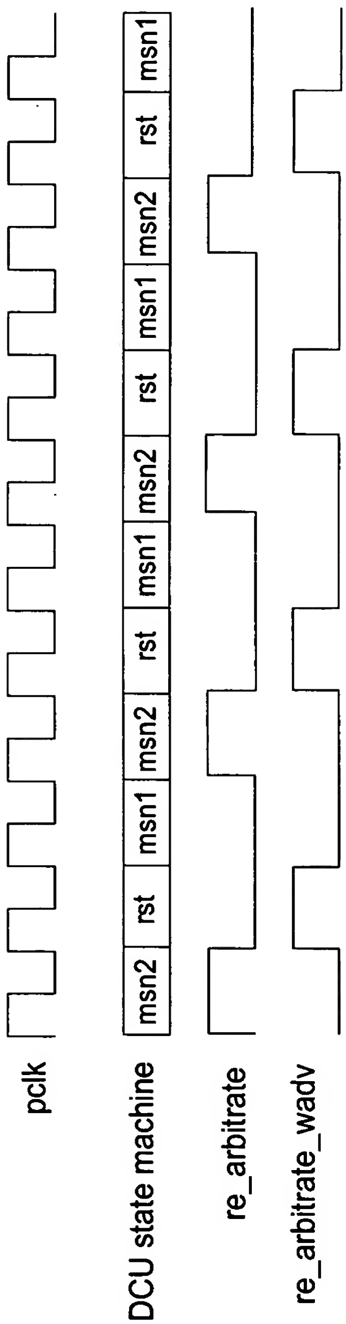

pclk
DCU state machine
msn2
rst
msn1
msn2
rst
msn1
msn2
rst
msn1
msn2
rst
msn1
re_arbitrate
re_arbitrate_wadv

FIG. 117

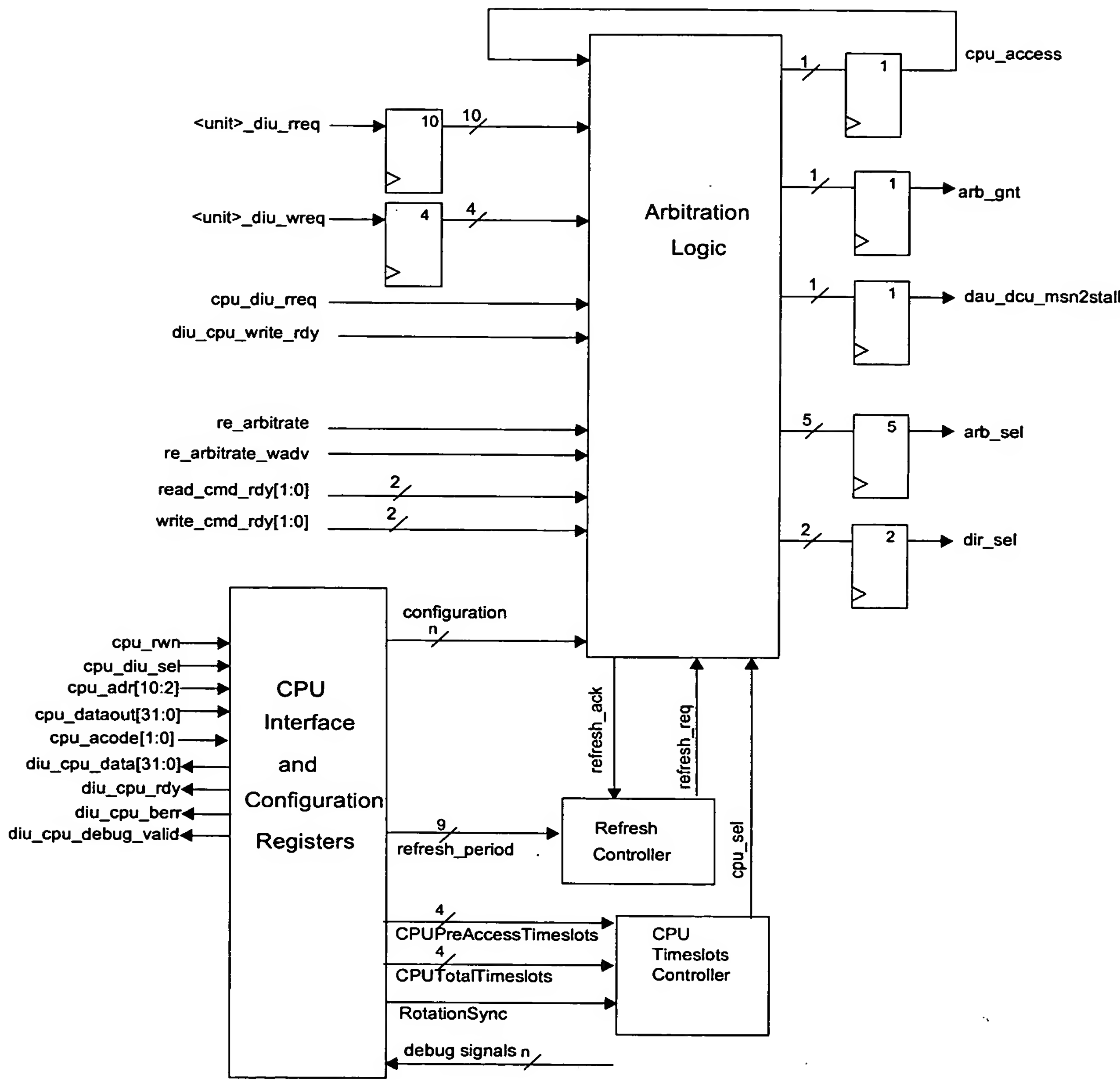

cpu_access
<unit>_diu_rreq
10
10
<unit>_diu_wreq
4
4
cpu_diu_rreq
diu_cpu_write_rdy
re_arbitrate
re_arbitrate_wadv
read_cmd_rdy[1:0]
2
write_cmd_rdy[1:0]
2
Arbitration Logic
1
1
arb_gnt
1
1
dau_dcu_msn2stall
1
1
5
5
arb_sel
2
2
dir_sel
configuration
n
cpu_rwn
cpu_diu_sel
cpu_adr[10:2]
cpu_dataout[31:0]
cpu_acode[1:0]
diu_cpu_data[31:0]
diu_cpu_rdy
diu_cpu_berr
diu_cpu_debug_valid
CPU Interface and Configuration Registers
refresh_ack
refresh_req
cpu_sel
9
refresh_period
Refresh Controller
4
CPUPreAccessTimeslots
4
CPUTotalTimeslots
RotationSync
debug signals n
CPU Timeslots Controller

FIG. 118

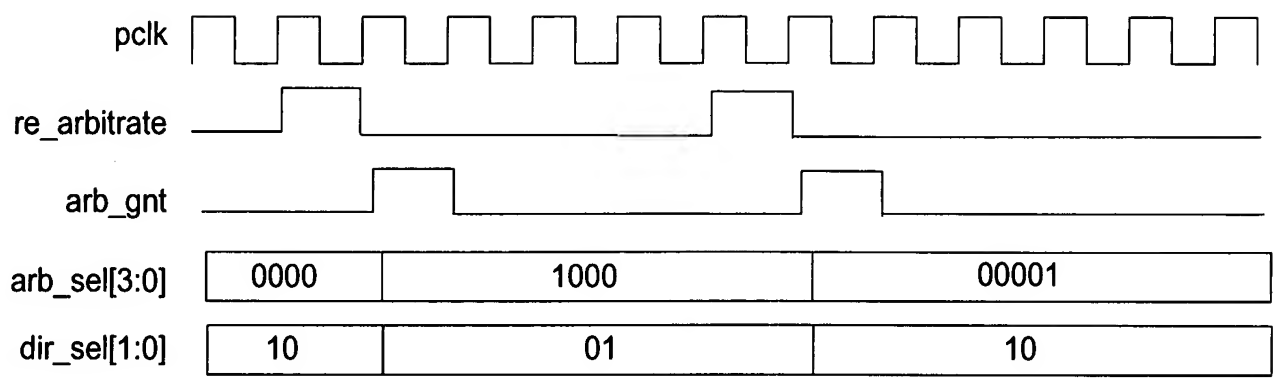

pclk
re_arbitrate
arb_gnt
arb_sel[3:0]
0000
1000
00001
dir_sel[1:0]
10
01
10

FIG. 119

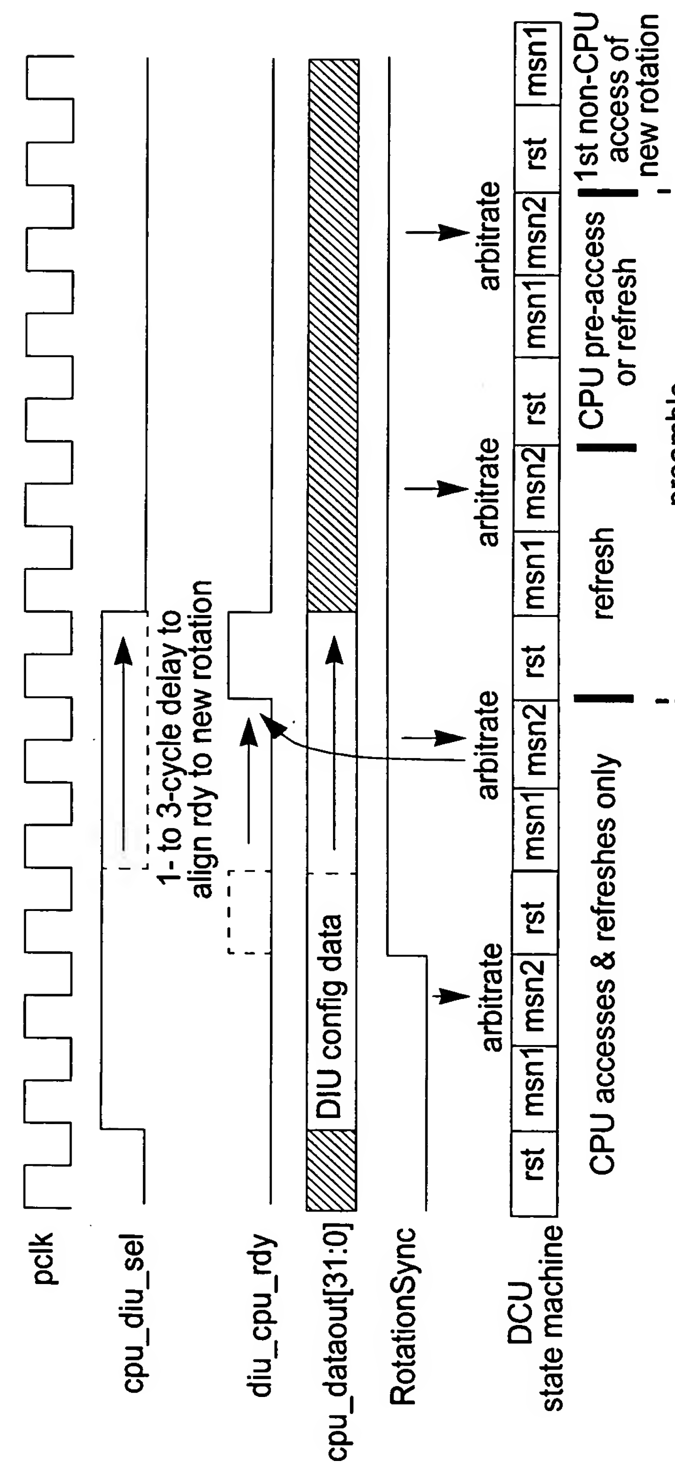

pclk
cpu_diu_sel
diu_cpu_rdy
cpu_dataout[31:0]
RotationSync
DCU
state machine
1- to 3-cycle delay to
align rdy to new rotation
DIU config data
arbitrate
arbitrate
arbitrate
arbitrate
rst
msn1
msn2
rst
msn1
msn2
rst
msn1
msn2
rst
msn1
msn2
rst
msn1
CPU accesses & refreshes only
refresh
CPU pre-access
or refresh
1st non-CPU
access of
new rotation
preamble

FIG. 120

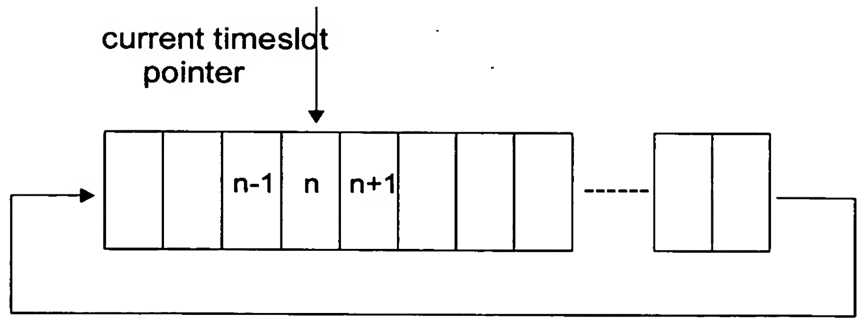


FIG. 121

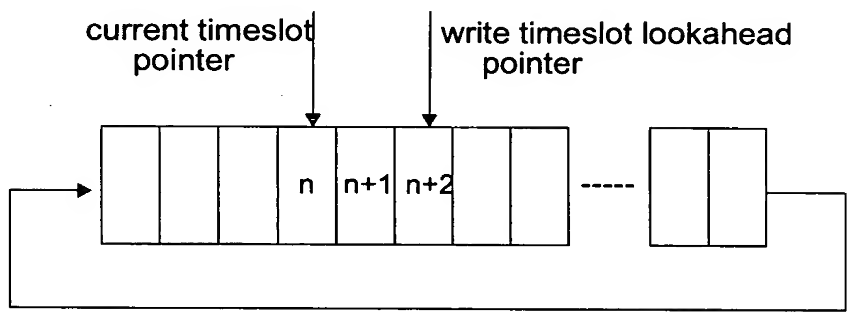


FIG. 122

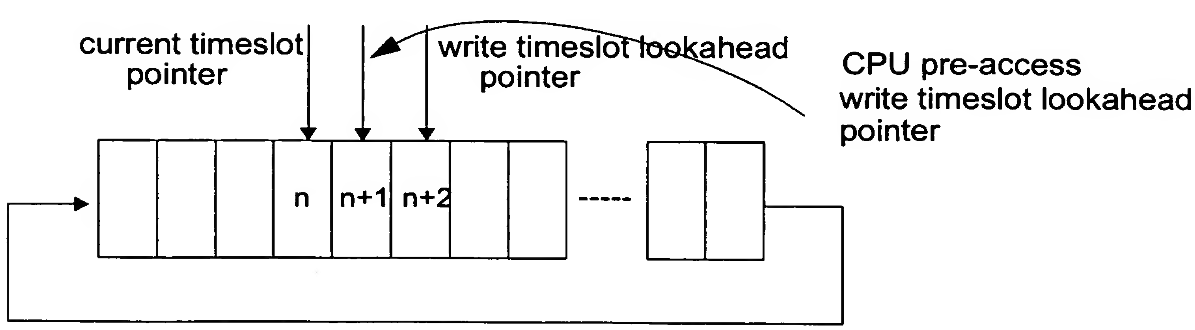


FIG. 123

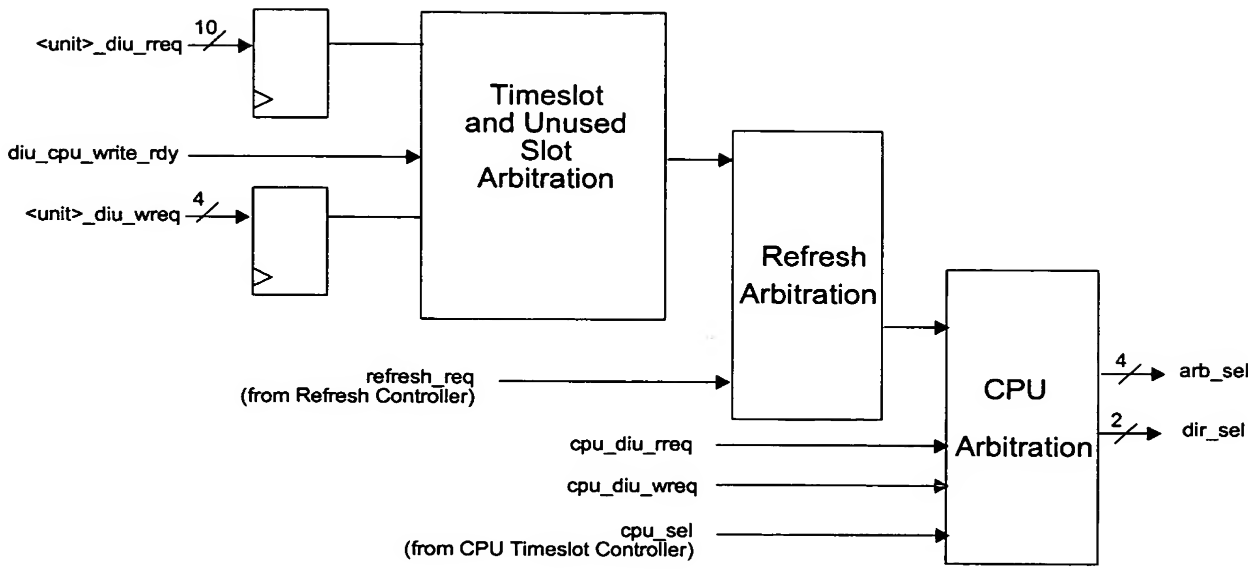


FIG. 124

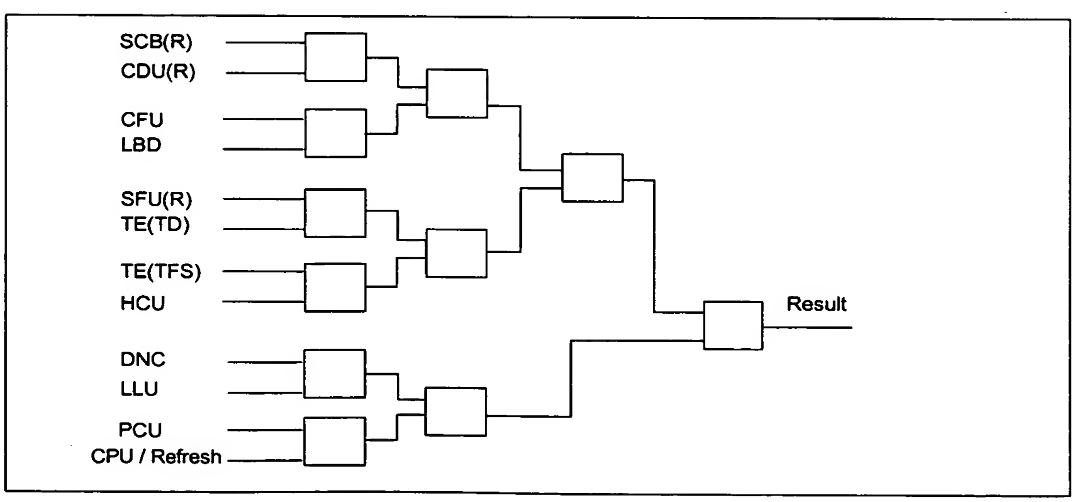


FIG. 125

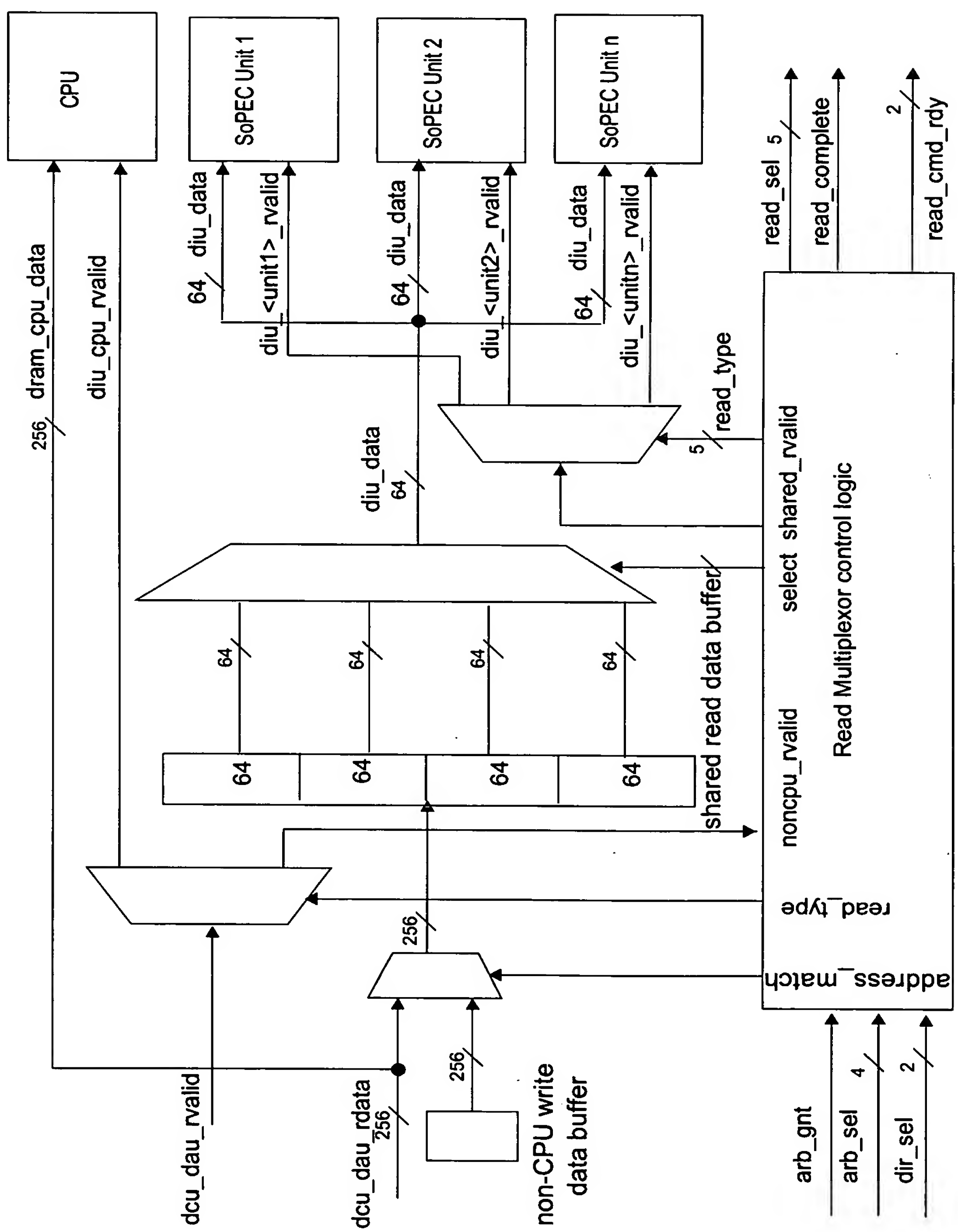

CPU
SoPEC Unit 1
SoPEC Unit 2
SoPEC Unit n
dram_cpu_data
256
diu_cpu_rvalid
64
diu_data
diu_<unit1>_rvalid
diu_<unit2>_rvalid
diu_<unitn>_rvalid
read_type
5
shared read data buffer
select
shared_rvalid
noncpu_rvalid
Read Multiplexor control logic
read_type
address_match
read_sel
read_complete
read_cmd_rdy
2
arb_gnt
arb_sel
4
dir_sel
dcu_dau_rvalid
dcu_dau_rdata
non-CPU write data buffer

FIG. 126

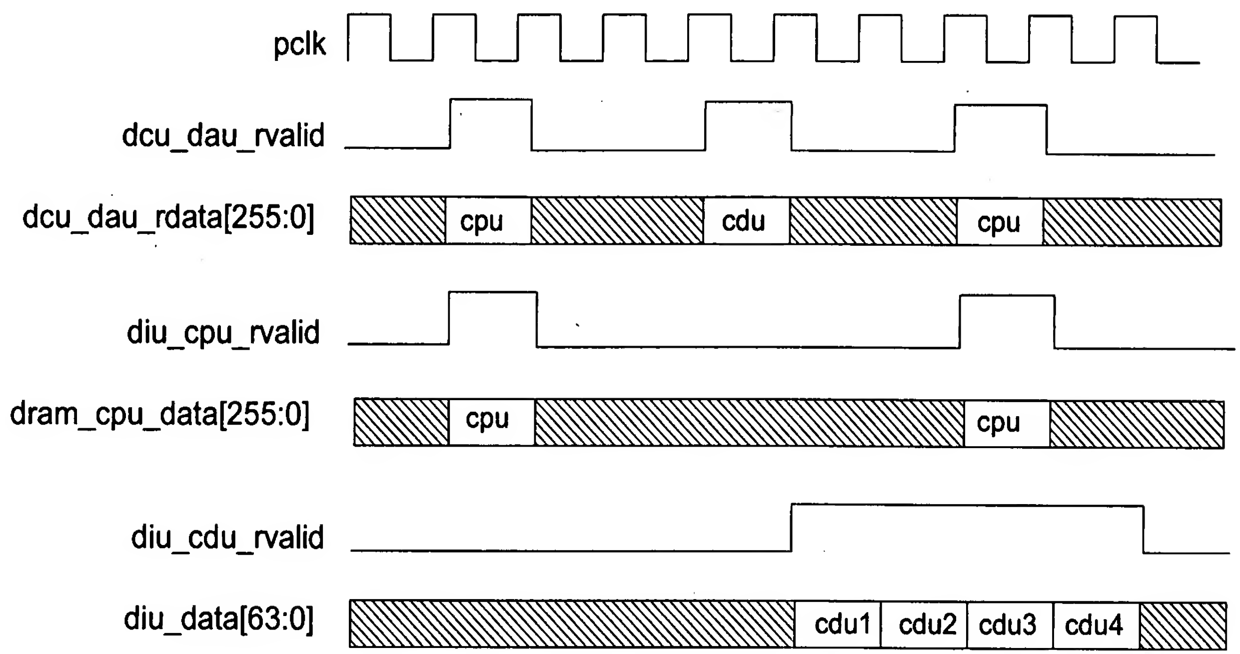

pclk
dcu_dau_rvalid
dcu_dau_rdata[255:0]
cpu
cdu
cpu
diu_cpu_rvalid
dram_cpu_data[255:0]
cpu
cpu
diu_cdu_rvalid
diu_data[63:0]
cdu1
cdu2
cdu3
cdu4

FIG. 127

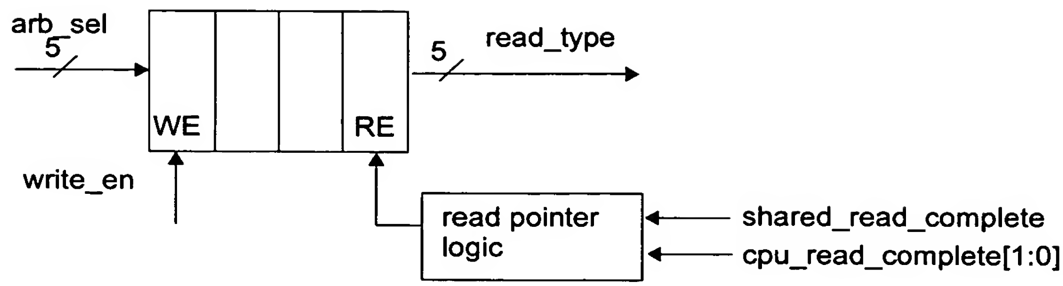


FIG. 128

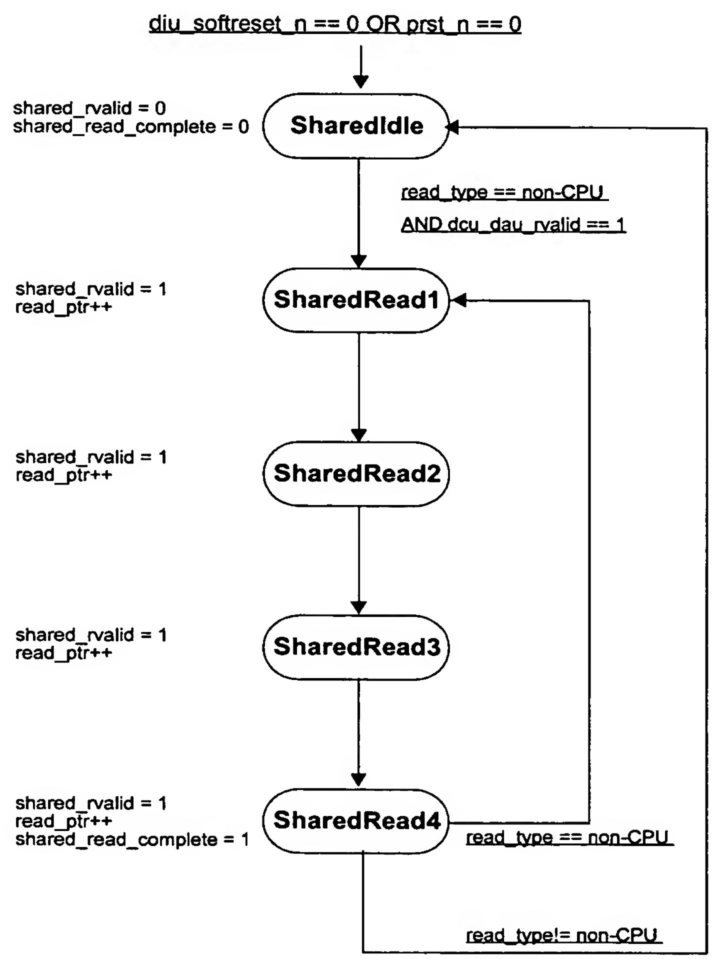


FIG. 129

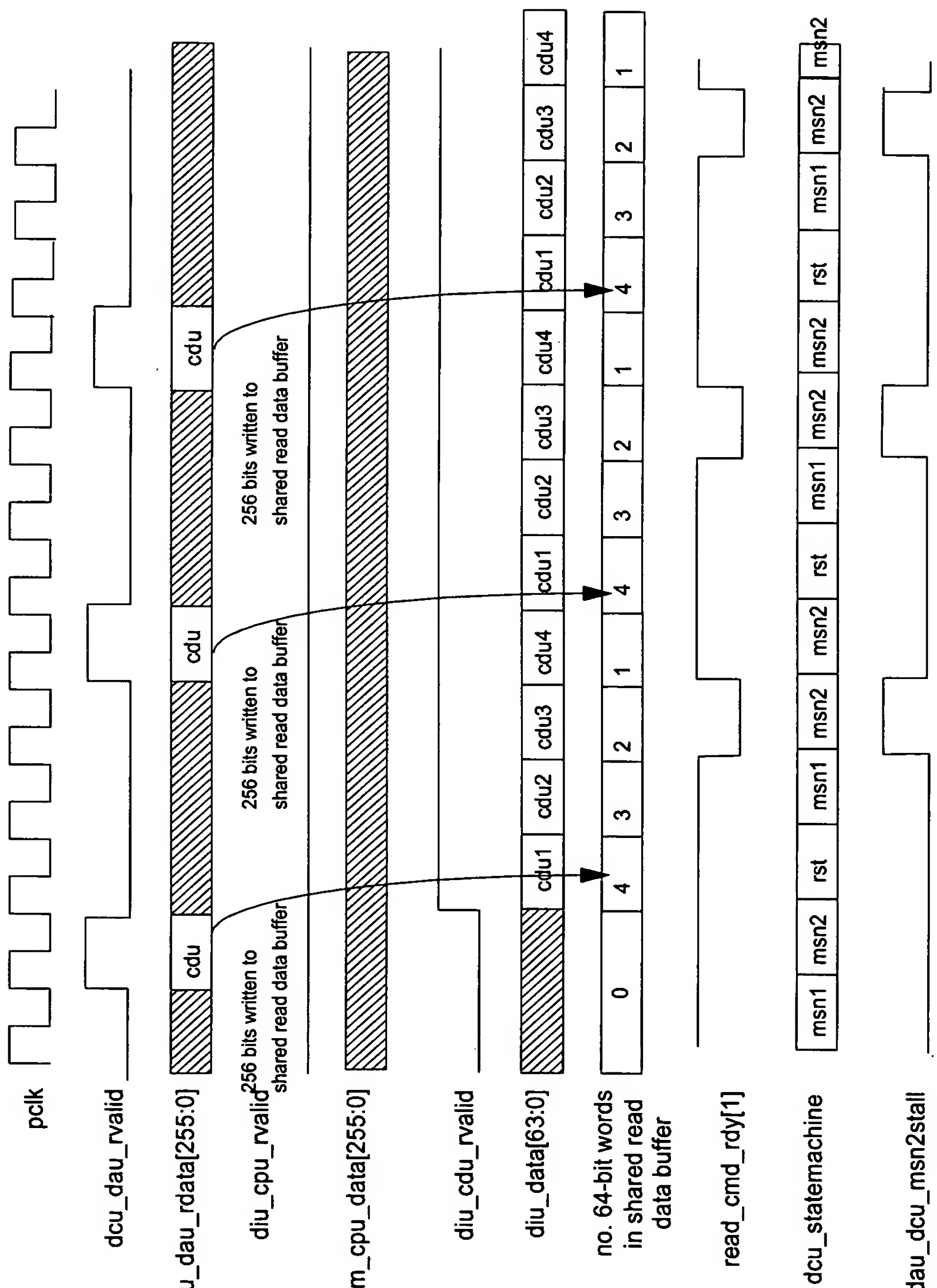

pclk
dcu_dau_rvalid
dcu_dau_rdata[255:0]
diu_cpu_rvalid
dram_cpu_data[255:0]
diu_cdu_rvalid
diu_data[63:0]
no. 64-bit words in shared read data buffer
read_cmd_rdy[1]
dcu_statemachine
dau_dcu_msn2stall
cdu
256 bits written to shared read data buffer
cdu1
cdu2
cdu3
cdu4
0
1
2
3
4
msn1
msn2
rst

FIG. 130

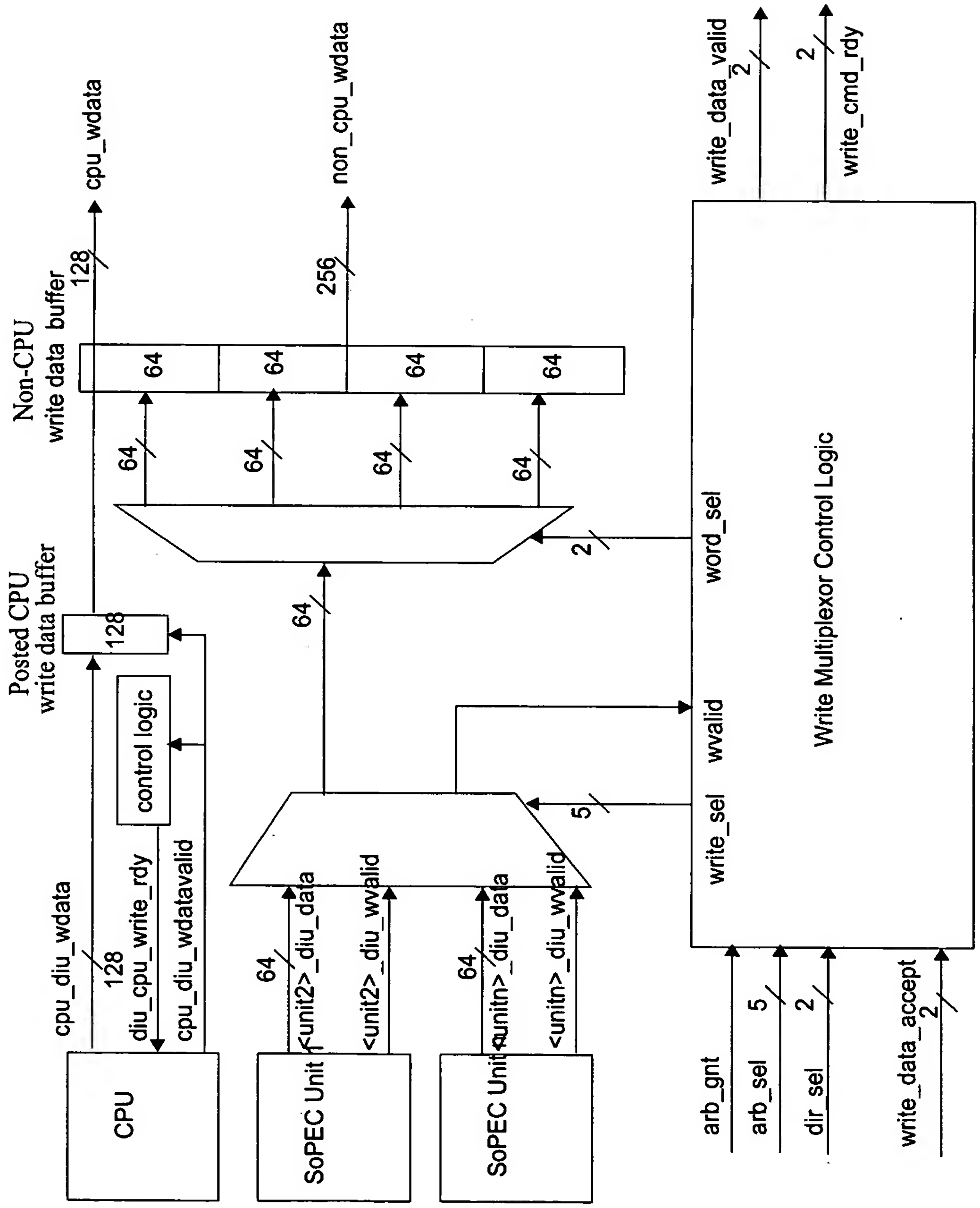

CPU
SoPEC Unit 1
SoPEC Unit n
cpu_diu_wdata
128
diu_cpu_write_rdy
cpu_diu_wdatavalid
control logic
Posted CPU write data buffer
128
cpu_wdata
Non-CPU write data buffer
64
64
64
64
256
non_cpu_wdata
64
<unit2>_diu_data
<unit2>_diu_wvalid
<unitn>_diu_data
<unitn>_diu_wvalid
5
2
write_sel
wvalid
word_sel
Write Multiplexor Control Logic
write_data_valid
write_cmd_rdy
arb_gnt
arb_sel
dir_sel
write_data_accept

FIG. 131

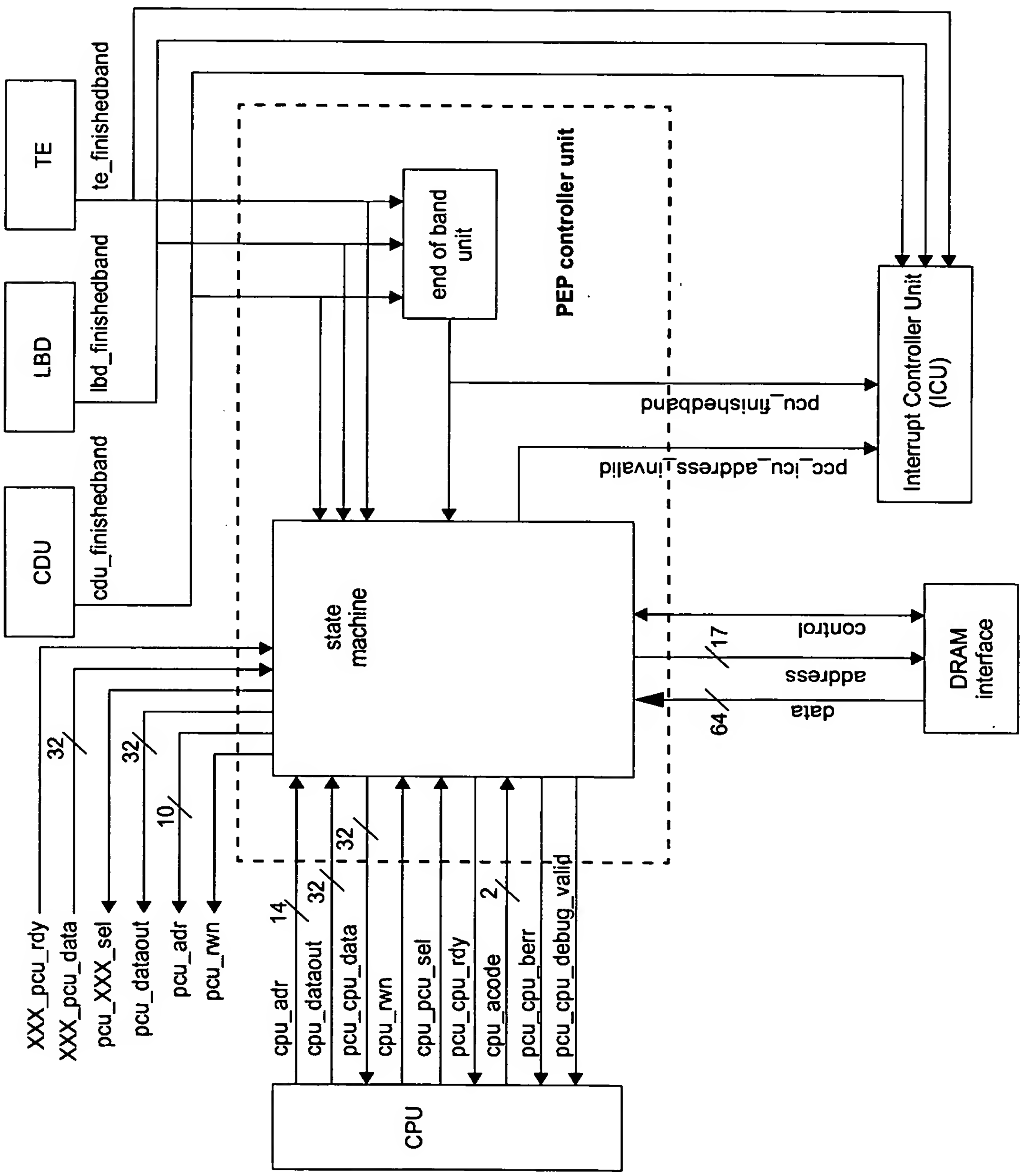

TE
LBD
CDU
te_finishedband
lbd_finishedband
cdu_finishedband
PEP controller unit
end of band unit
state machine
Interrupt Controller Unit (ICU)
pcu_finishedband
pcc_icu_address_invalid
DRAM interface
control
address
data
17
64
XXX_pcu_rdy
XXX_pcu_data
pcu_XXX_sel
pcu_dataout
pcu_adr
pcu_rwn
32
32
10
cpu_adr
cpu_dataout
pcu_cpu_data
cpu_rwn
cpu_pcu_sel
pcu_cpu_rdy
cpu_acode
pcu_cpu_berr
pcu_cpu_debug_valid
14
32
32
2
CPU

FIG. 132

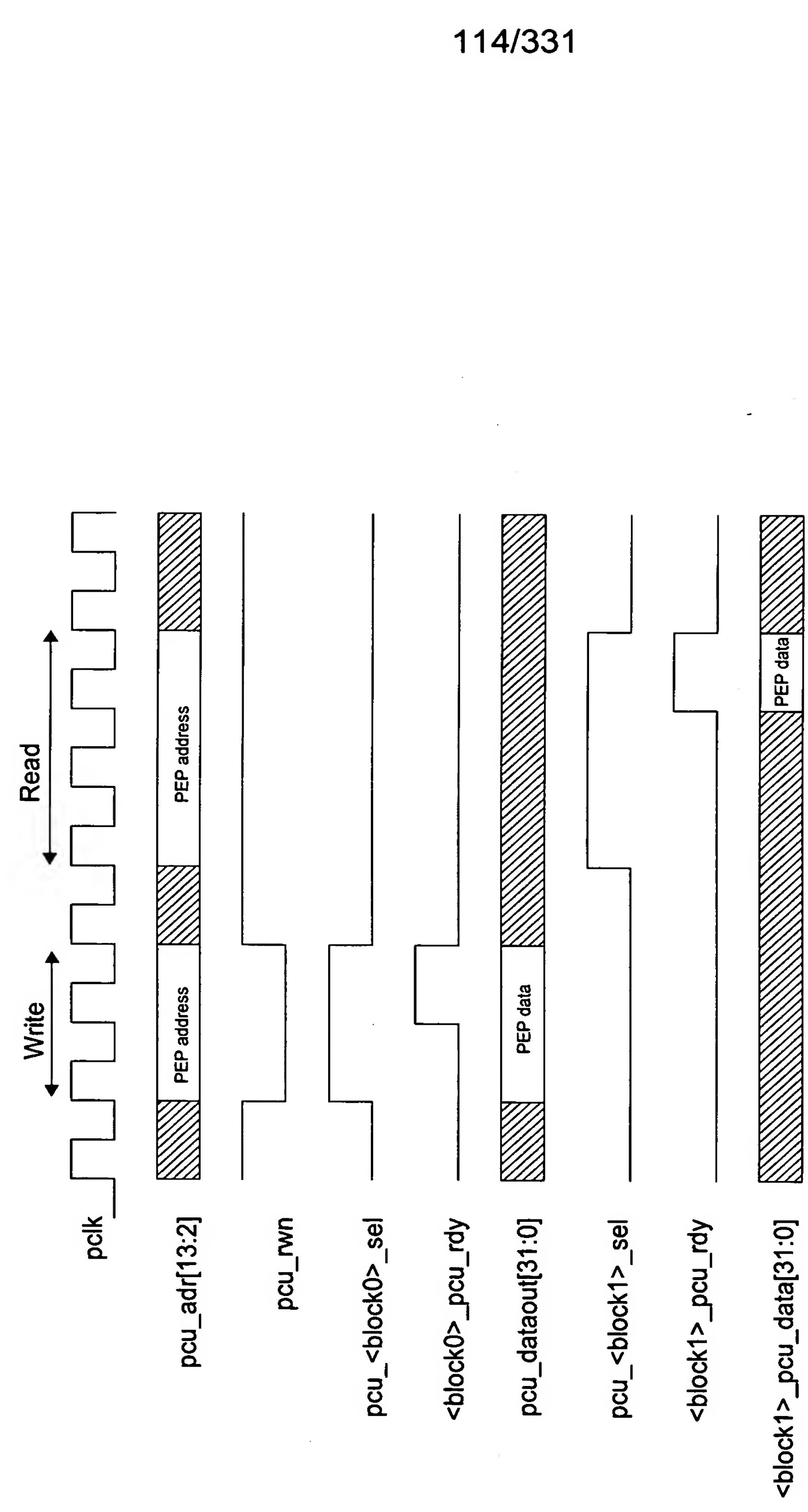

Write
Read
pclk
pcu_adr[13:2]
PEP address
PEP address
pcu_rwn
pcu_<block0>_sel
<block0>_pcu_rdy
pcu_dataout[31:0]
PEP data
pcu_<block1>_sel
<block1>_pcu_rdy
<block1>_pcu_data[31:0]
PEP data

FIG. 133

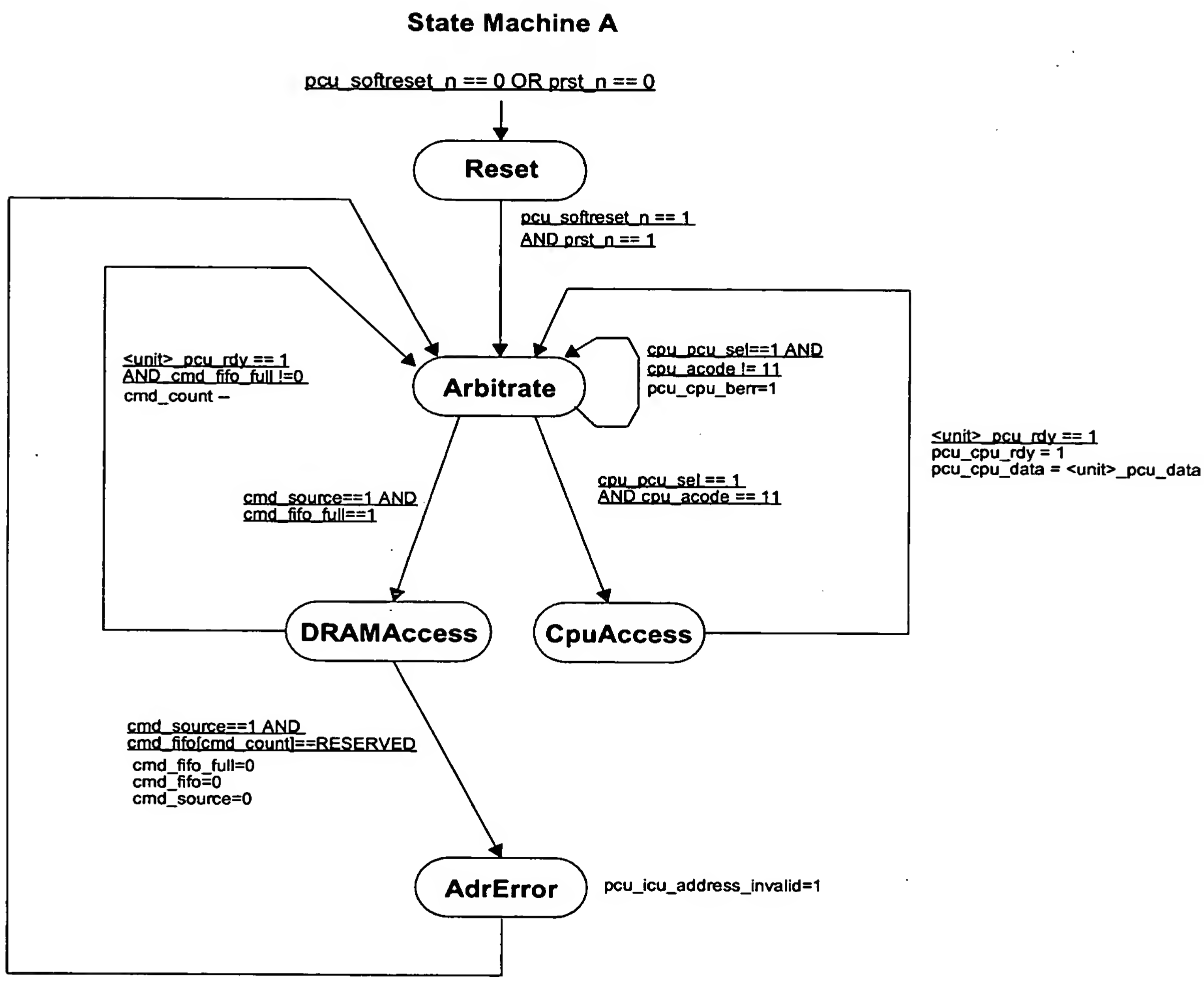

State Machine A
pcu_softreset_n == 0 OR prst_n == 0
Reset
pcu_softreset_n == 1
AND prst_n == 1
Arbitrate
<unit>_pcu_rdy == 1
AND cmd_fifo_full !=0
cmd_count --
cpu_pcu_sel==1 AND
cpu_acode != 11
pcu_cpu_berr=1
<unit>_pcu_rdy == 1
pcu_cpu_rdy = 1
pcu_cpu_data = <unit>_pcu_data
cmd_source==1 AND
cmd_fifo_full==1
cpu_pcu_sel == 1
AND cpu_acode == 11
DRAMAccess
CpuAccess
cmd_source==1 AND
cmd_fifo[cmd_count]==RESERVED
cmd_fifo_full=0
cmd_fifo=0
cmd_source=0
AdrError
pcu_icu_address_invalid=1

FIG. 134

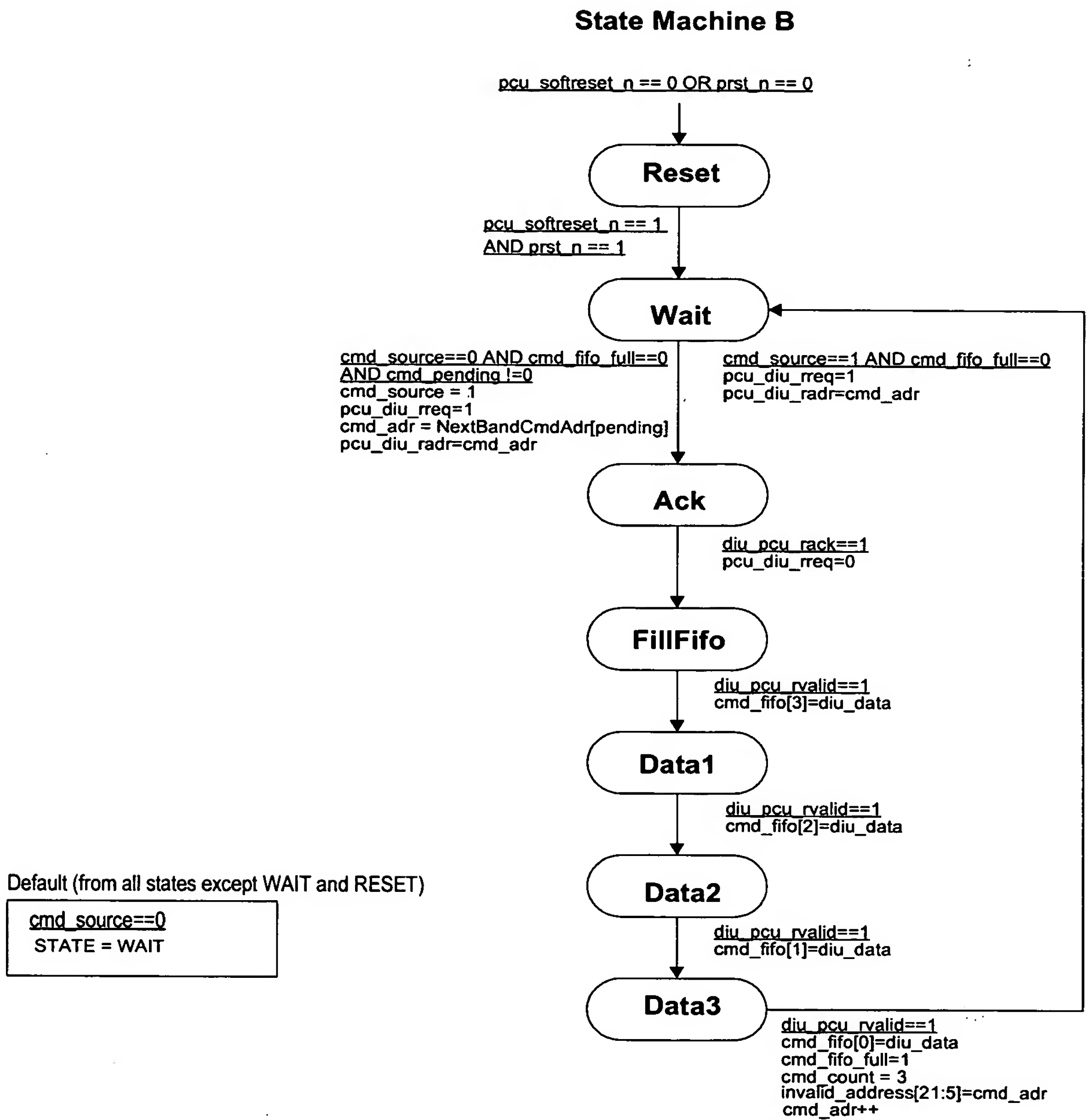

State Machine B
pcu_softreset_n == 0 OR prst_n == 0
Reset
pcu_softreset_n == 1
AND prst_n == 1
Wait
cmd_source==0 AND cmd_fifo_full==0
AND cmd_pending !=0
cmd_source = 1
pcu_diu_rreq=1
cmd_adr = NextBandCmdAdr[pending]
pcu_diu_radr=cmd_adr
cmd_source==1 AND cmd_fifo_full==0
pcu_diu_rreq=1
pcu_diu_radr=cmd_adr
Ack
diu_pcu_rack==1
pcu_diu_rreq=0
FillFifo
diu_pcu_rvalid==1
cmd_fifo[3]=diu_data
Data1
diu_pcu_rvalid==1
cmd_fifo[2]=diu_data
Data2
diu_pcu_rvalid==1
cmd_fifo[1]=diu_data
Data3
diu_pcu_rvalid==1
cmd_fifo[0]=diu_data
cmd_fifo_full=1
cmd_count = 3
invalid_address[21:5]=cmd_adr
cmd_adr++
Default (from all states except WAIT and RESET)
cmd_source==0
STATE = WAIT

FIG. 135

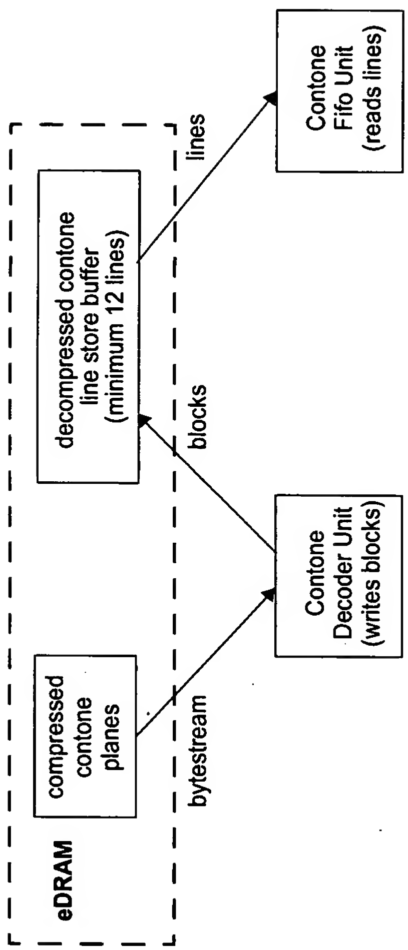

eDRAM
compressed contone planes
decompressed contone line store buffer (minimum 12 lines)
bytestream
blocks
lines
Contone Decoder Unit (writes blocks)
Contone Fifo Unit (reads lines)

FIG. 136

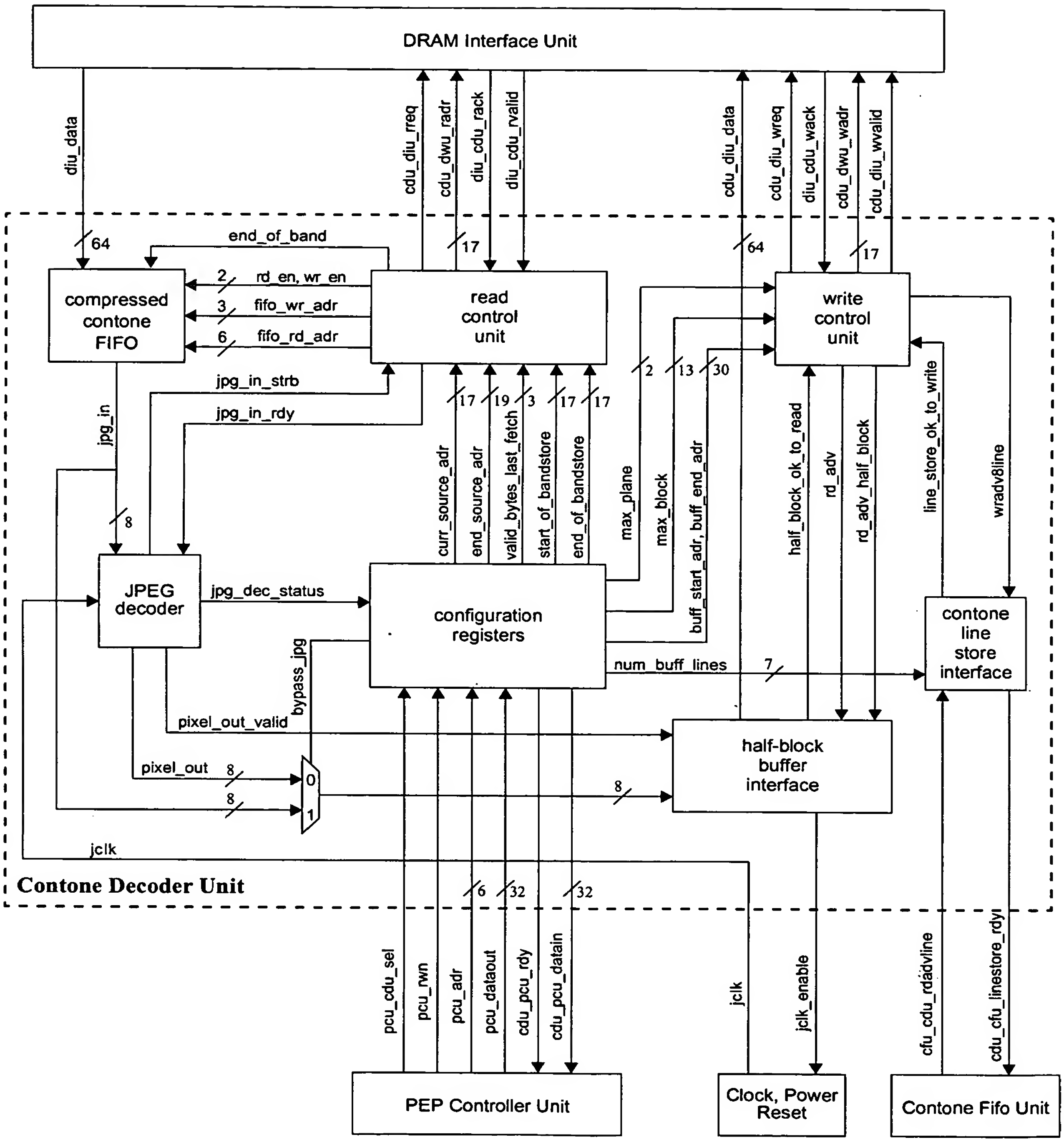

DRAM Interface Unit
diu_data
cdu_diu_rreq
cdu_dwu_radr
diu_cdu_rack
diu_cdu_rvalid
cdu_diu_data
cdu_diu_wreq
diu_cdu_wack
cdu_dwu_wadr
cdu_diu_wvalid
64
17
64
17
end_of_band
compressed contone FIFO
rd_en, wr_en
fifo_wr_adr
fifo_rd_adr
read control unit
write control unit
jpg_in_strb
jpg_in_rdy
jpg_in
curr_source_adr
end_source_adr
valid_bytes_last_fetch
start_of_bandstore
end_of_bandstore
max_plane
max_block
buff_start_adr, buff_end_adr
half_block_ok_to_read
rd_adv
rd_adv_half_block
line_store_ok_to_write
wradv8line
JPEG decoder
jpg_dec_status
configuration registers
bypass_jpg
num_buff_lines
contone line store interface
pixel_out_valid
half-block buffer interface
pixel_out
jclk
Contone Decoder Unit
pcu_cdu_sel
pcu_rwn
pcu_adr
pcu_dataout
cdu_pcu_rdy
cdu_pcu_datain
jclk
jclk_enable
cfu_cdu_rdadvline
cdu_cfu_linestore_rdy
PEP Controller Unit
Clock, Power Reset
Contone Fifo Unit

FIG. 137

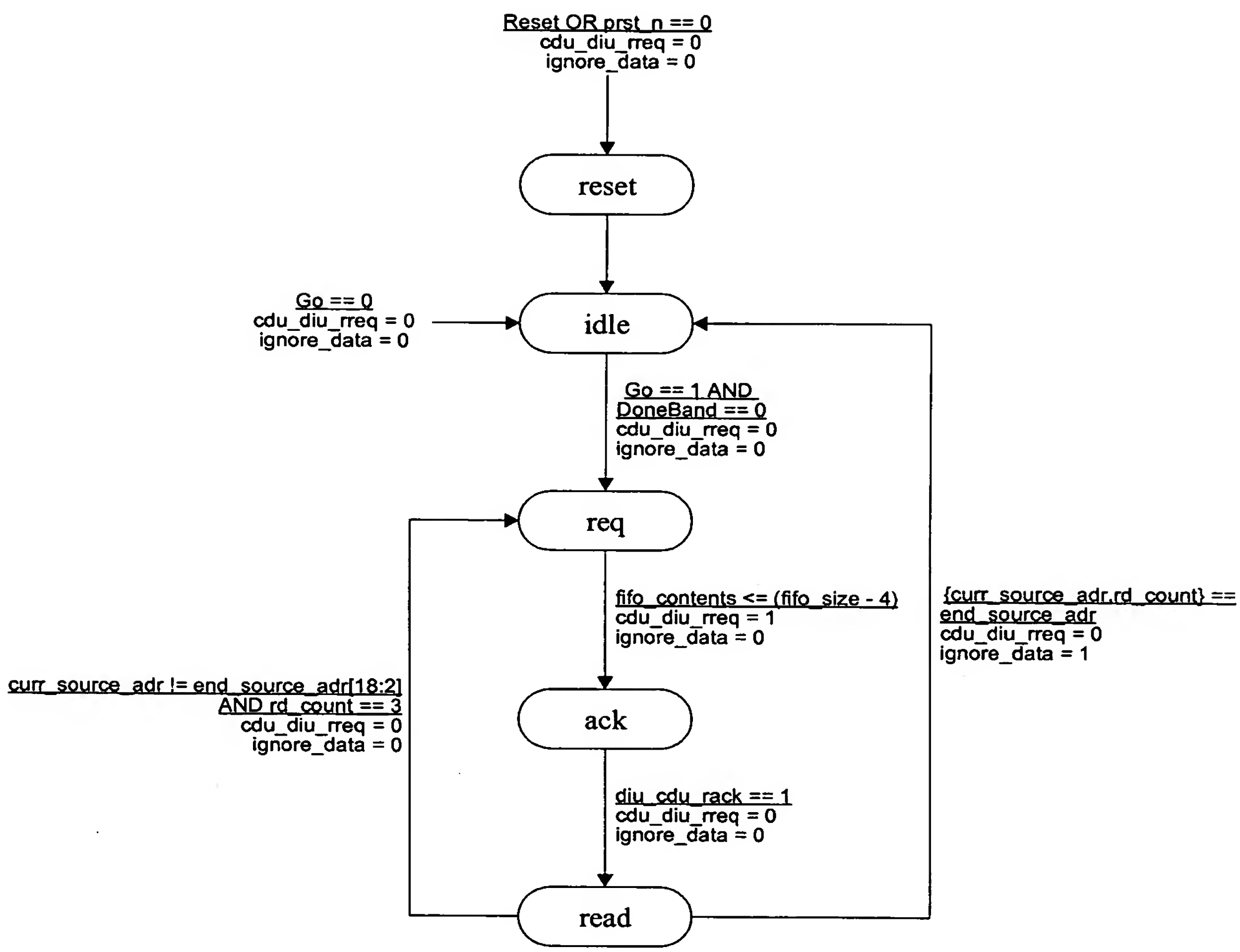
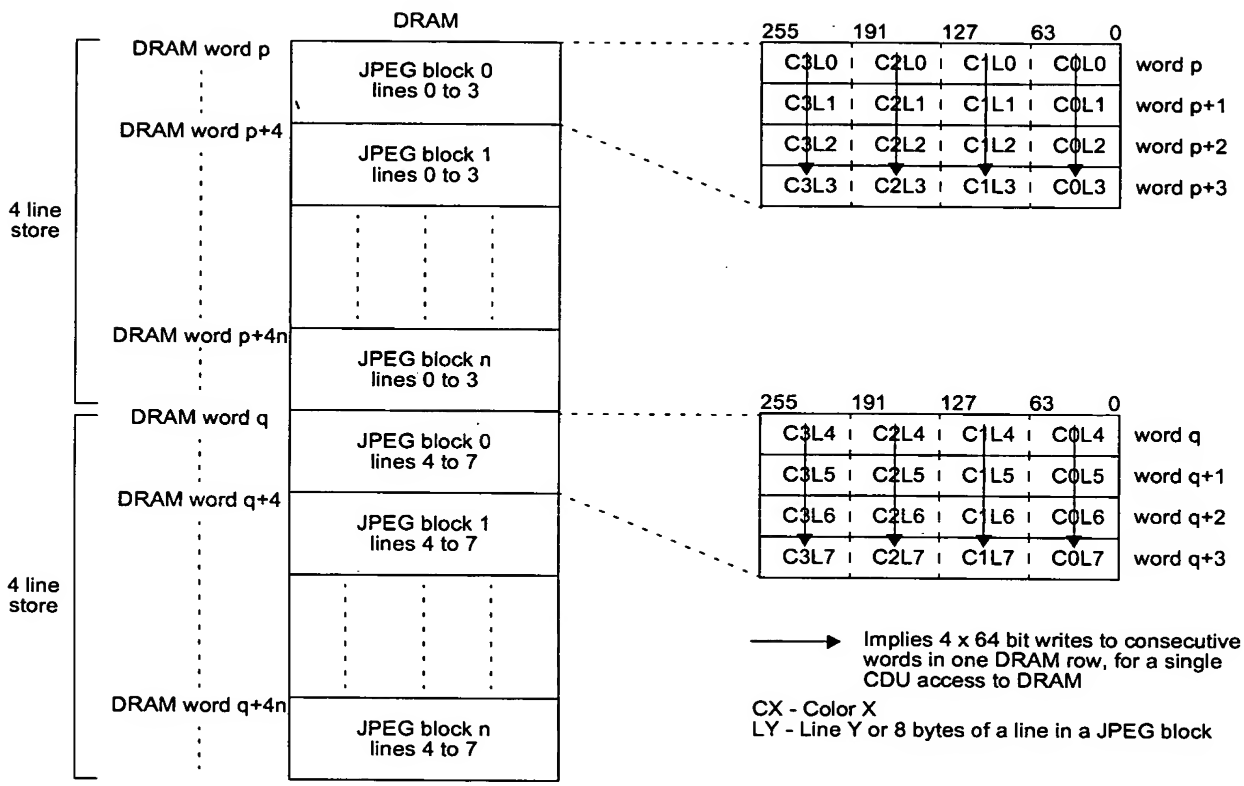

Reset OR prst_n == 0
cdu_diu_rreq = 0
ignore_data = 0
reset
Go == 0
cdu_diu_rreq = 0
ignore_data = 0
idle
Go == 1 AND
DoneBand == 0
cdu_diu_rreq = 0
ignore_data = 0
req
fifo_contents <= (fifo_size - 4)
cdu_diu_rreq = 1
ignore_data = 0
{curr_source_adr,rd_count} ==
end_source_adr
cdu_diu_rreq = 0
ignore_data = 1
curr_source_adr != end_source_adr[18:2]
AND rd_count == 3
cdu_diu_rreq = 0
ignore_data = 0
ack
diu_cdu_rack == 1
cdu_diu_rreq = 0
ignore_data = 0
read

FIG. 138



DRAM
DRAM word p
DRAM word p+4
DRAM word p+4n
DRAM word q
DRAM word q+4
DRAM word q+4n
4 line store
4 line store
JPEG block 0 lines 0 to 3
JPEG block 1 lines 0 to 3
JPEG block n lines 0 to 3
JPEG block 0 lines 4 to 7
JPEG block 1 lines 4 to 7
JPEG block n lines 4 to 7
255 191 127 63 0
C3L0 C2L0 C1L0 C0L0 word p
C3L1 C2L1 C1L1 C0L1 word p+1
C3L2 C2L2 C1L2 C0L2 word p+2
C3L3 C2L3 C1L3 C0L3 word p+3
255 191 127 63 0
C3L4 C2L4 C1L4 C0L4 word q
C3L5 C2L5 C1L5 C0L5 word q+1
C3L6 C2L6 C1L6 C0L6 word q+2
C3L7 C2L7 C1L7 C0L7 word q+3
Implies 4 x 64 bit writes to consecutive words in one DRAM row, for a single CDU access to DRAM
CX - Color X
LY - Line Y or 8 bytes of a line in a JPEG block

FIG. 139

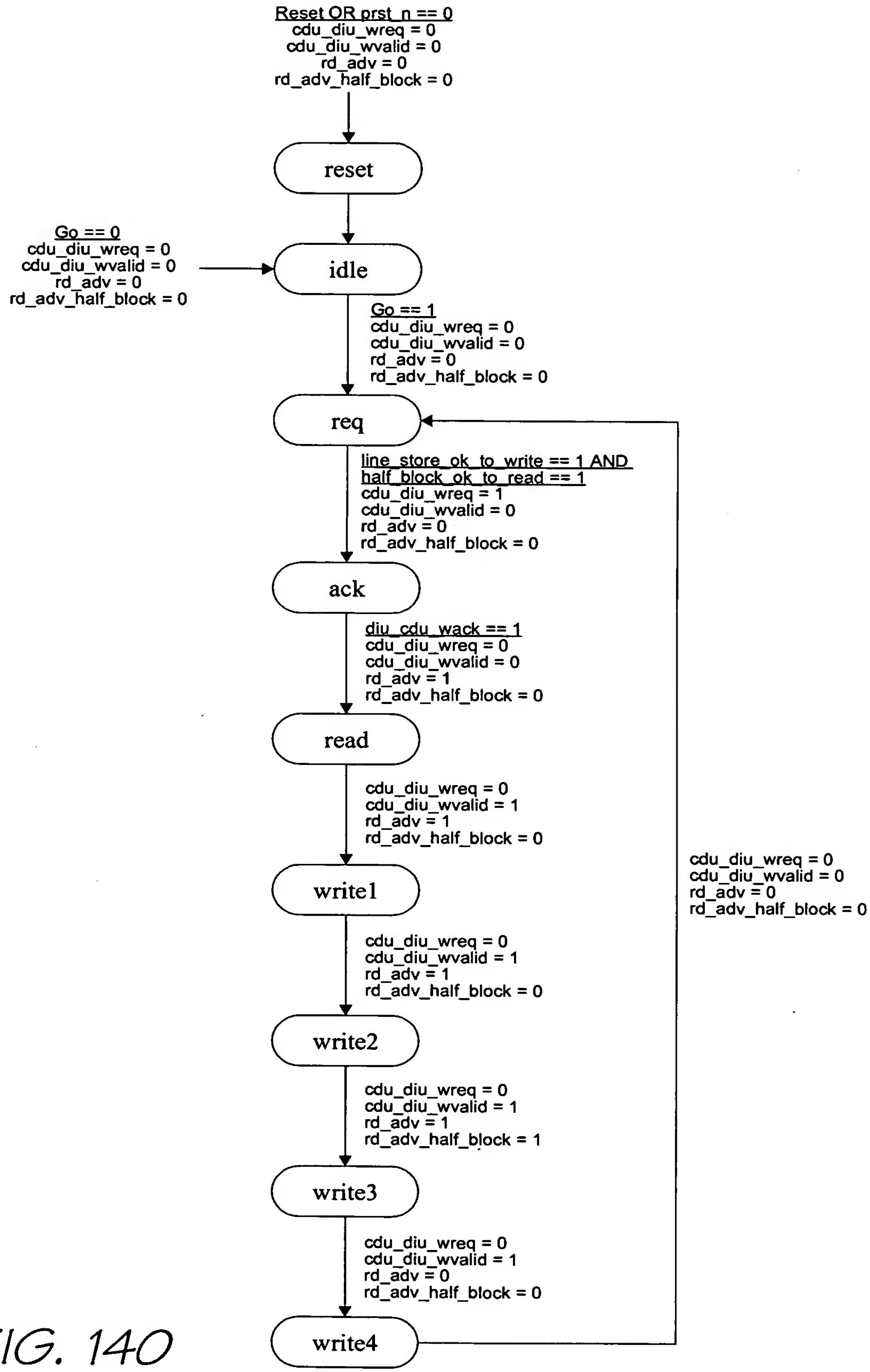

Reset OR prst_n == 0
cdu_diu_wreq = 0
cdu_diu_wvalid = 0
rd_adv = 0
rd_adv_half_block = 0
reset
Go == 0
cdu_diu_wreq = 0
cdu_diu_wvalid = 0
rd_adv = 0
rd_adv_half_block = 0
idle
Go == 1
cdu_diu_wreq = 0
cdu_diu_wvalid = 0
rd_adv = 0
rd_adv_half_block = 0
req
line_store_ok_to_write == 1 AND
half_block_ok_to_read == 1
cdu_diu_wreq = 1
cdu_diu_wvalid = 0
rd_adv = 0
rd_adv_half_block = 0
ack
diu_cdu_wack == 1
cdu_diu_wreq = 0
cdu_diu_wvalid = 0
rd_adv = 1
rd_adv_half_block = 0
read
cdu_diu_wreq = 0
cdu_diu_wvalid = 1
rd_adv = 1
rd_adv_half_block = 0
write1
cdu_diu_wreq = 0
cdu_diu_wvalid = 1
rd_adv = 1
rd_adv_half_block = 0
write2
cdu_diu_wreq = 0
cdu_diu_wvalid = 1
rd_adv = 1
rd_adv_half_block = 1
write3
cdu_diu_wreq = 0
cdu_diu_wvalid = 1
rd_adv = 0
rd_adv_half_block = 0
write4
cdu_diu_wreq = 0
cdu_diu_wvalid = 0
rd_adv = 0
rd_adv_half_block = 0

FIG. 140

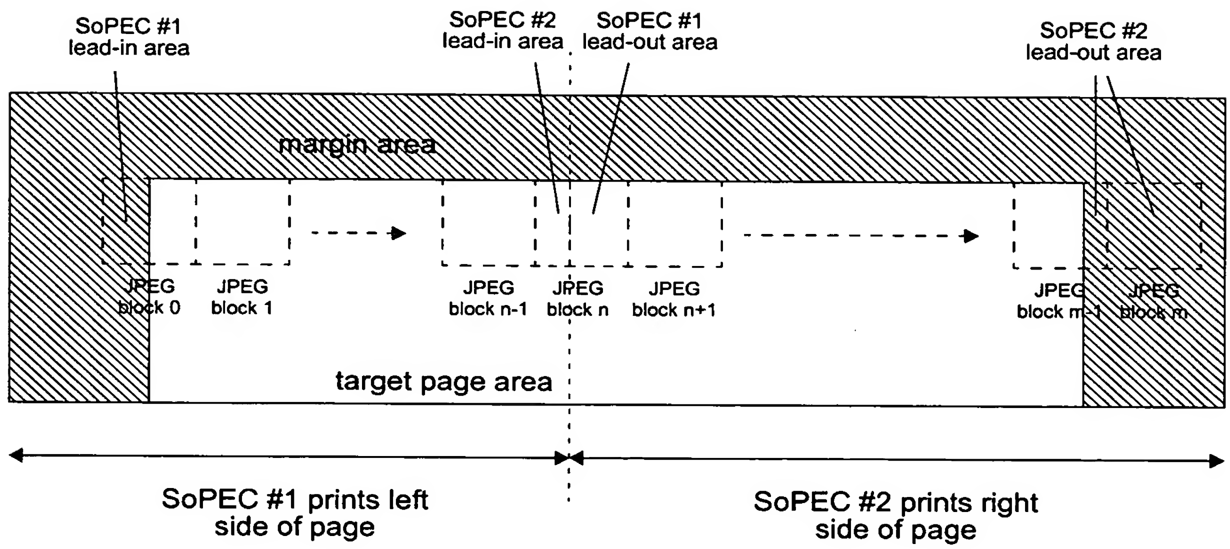

SoPEC #1
lead-in area
SoPEC #2
lead-in area
SoPEC #1
lead-out area
SoPEC #2
lead-out area
margin area
JPEG
block 0
JPEG
block 1
JPEG
block n-1
JPEG
block n
JPEG
block n+1
JPEG
block m-1
JPEG
block m
target page area
SoPEC #1 prints left
side of page
SoPEC #2 prints right
side of page

FIG. 141

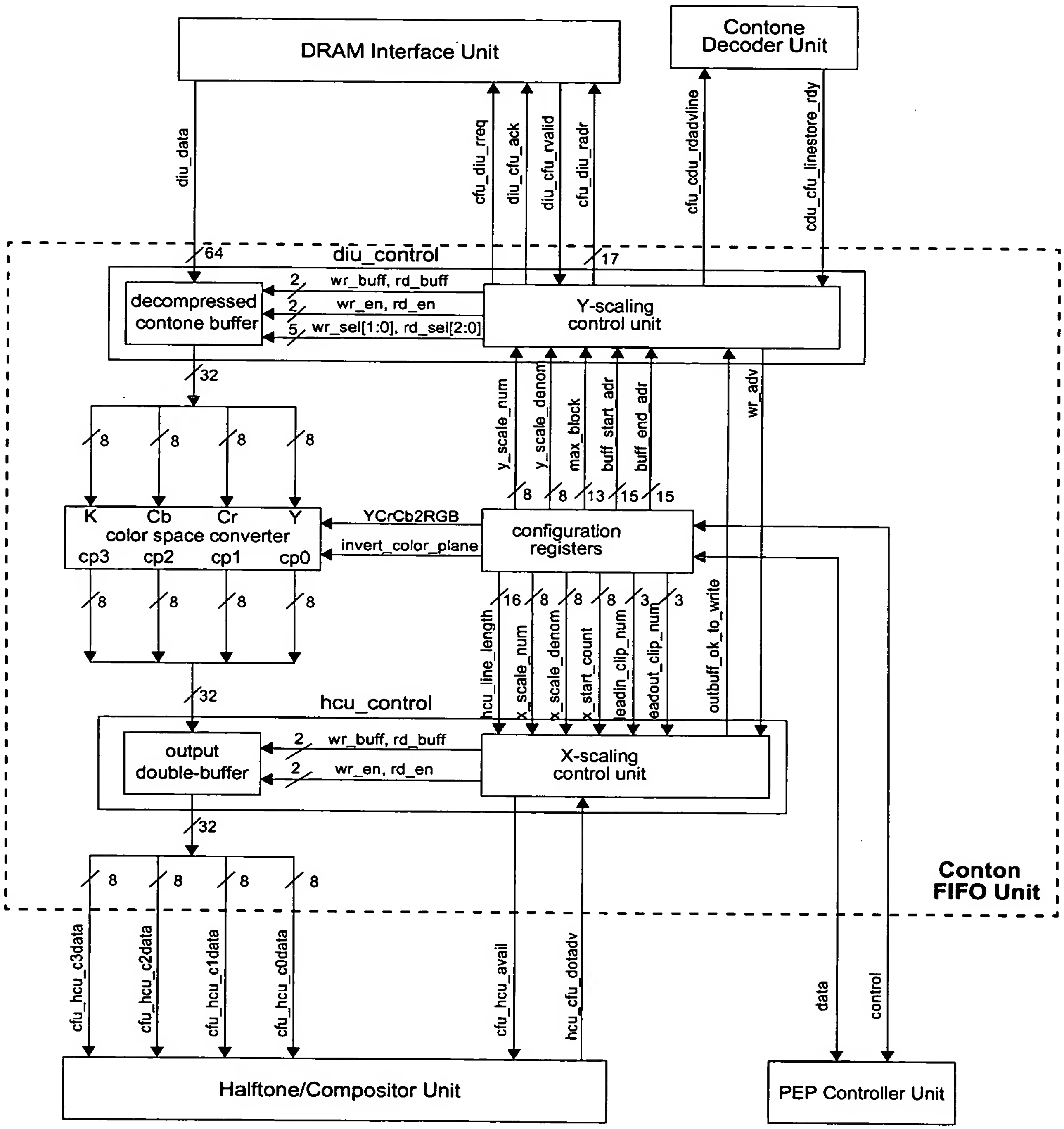

DRAM Interface Unit
Contone Decoder Unit
diu_data
cfu_diu_rreq
diu_cfu_ack
diu_cfu_rvalid
cfu_diu_radr
cfu_cdu_rdadvline
cdu_cfu_linestore_rdy
64
diu_control
17
decompressed contone buffer
2 wr_buff, rd_buff
2 wr_en, rd_en
5 wr_sel[1:0], rd_sel[2:0]
Y-scaling control unit
32
y_scale_num
y_scale_denom
max_block
buff_start_adr
buff_end_adr
wr_adv
8 8 13 15 15
8 8 8 8
K Cb Cr Y
color space converter
cp3 cp2 cp1 cp0
YCrCb2RGB
invert_color_plane
configuration registers
16 8 8 8 3 3
hcu_line_length
x_scale_num
x_scale_denom
x_start_count
leadin_clip_num
leadout_clip_num
outbuff_ok_to_write
8 8 8 8
32
hcu_control
output double-buffer
2 wr_buff, rd_buff
2 wr_en, rd_en
X-scaling control unit
32
8 8 8 8
Conton FIFO Unit
cfu_hcu_c3data
cfu_hcu_c2data
cfu_hcu_c1data
cfu_hcu_c0data
cfu_hcu_avail
hcu_cfu_dotadv
data
control
Halftone/Compositor Unit
PEP Controller Unit

FIG. 142

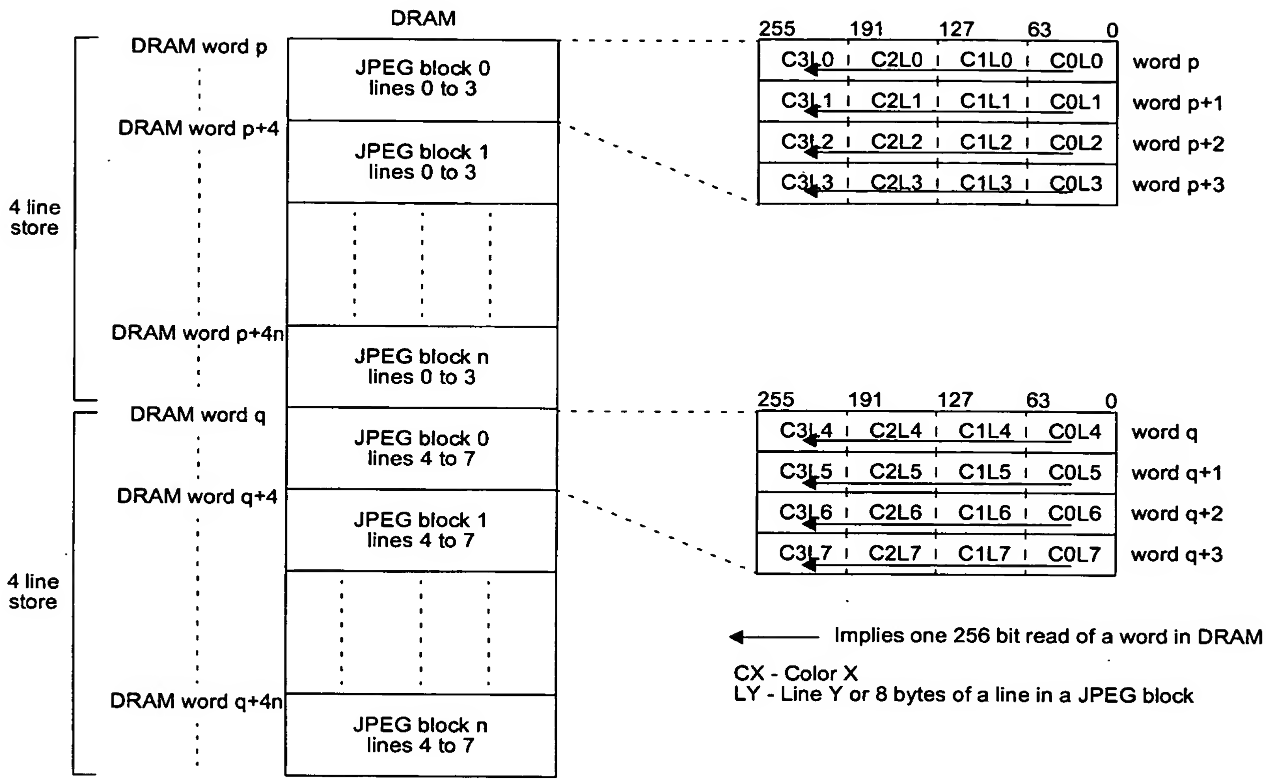

DRAM
DRAM word p
DRAM word p+4
DRAM word p+4n
DRAM word q
DRAM word q+4
DRAM word q+4n
4 line store
4 line store
JPEG block 0 lines 0 to 3
JPEG block 1 lines 0 to 3
JPEG block n lines 0 to 3
JPEG block 0 lines 4 to 7
JPEG block 1 lines 4 to 7
JPEG block n lines 4 to 7
255 191 127 63 0
C3L0 C2L0 C1L0 C0L0 word p
C3L1 C2L1 C1L1 C0L1 word p+1
C3L2 C2L2 C1L2 C0L2 word p+2
C3L3 C2L3 C1L3 C0L3 word p+3
255 191 127 63 0
C3L4 C2L4 C1L4 C0L4 word q
C3L5 C2L5 C1L5 C0L5 word q+1
C3L6 C2L6 C1L6 C0L6 word q+2
C3L7 C2L7 C1L7 C0L7 word q+3
Implies one 256 bit read of a word in DRAM
CX - Color X
LY - Line Y or 8 bytes of a line in a JPEG block

FIG. 143

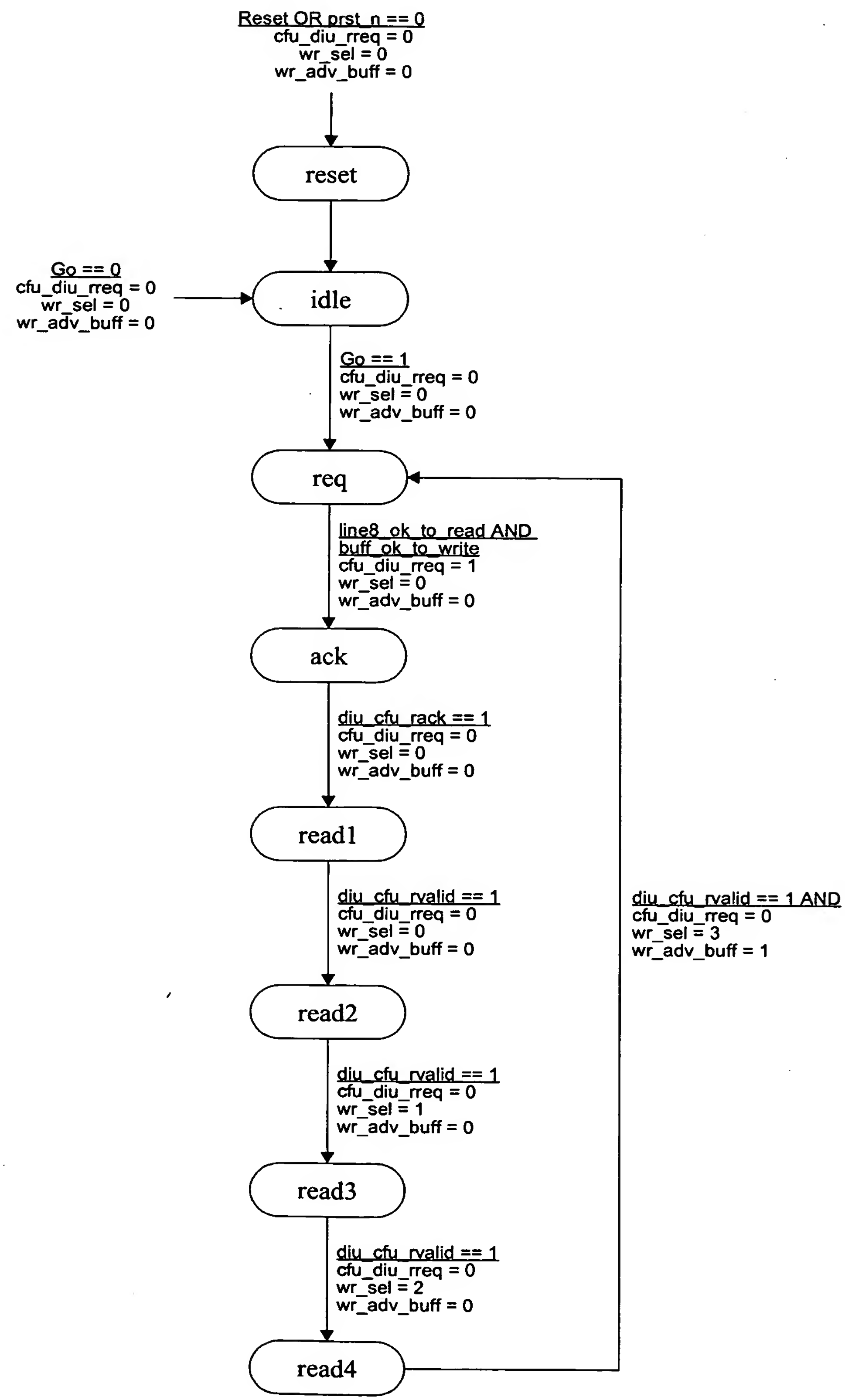

Reset OR prst_n == 0
cfu_diu_rreq = 0
wr_sel = 0
wr_adv_buff = 0
reset
Go == 0
cfu_diu_rreq = 0
wr_sel = 0
wr_adv_buff = 0
idle
Go == 1
cfu_diu_rreq = 0
wr_sel = 0
wr_adv_buff = 0
req
line8_ok_to_read AND
buff_ok_to_write
cfu_diu_rreq = 1
wr_sel = 0
wr_adv_buff = 0
ack
diu_cfu_rack == 1
cfu_diu_rreq = 0
wr_sel = 0
wr_adv_buff = 0
read1
diu_cfu_rvalid == 1
cfu_diu_rreq = 0
wr_sel = 0
wr_adv_buff = 0
read2
diu_cfu_rvalid == 1
cfu_diu_rreq = 0
wr_sel = 1
wr_adv_buff = 0
read3
diu_cfu_rvalid == 1
cfu_diu_rreq = 0
wr_sel = 2
wr_adv_buff = 0
read4
diu_cfu_rvalid == 1 AND
cfu_diu_rreq = 0
wr_sel = 3
wr_adv_buff = 1

FIG. 144

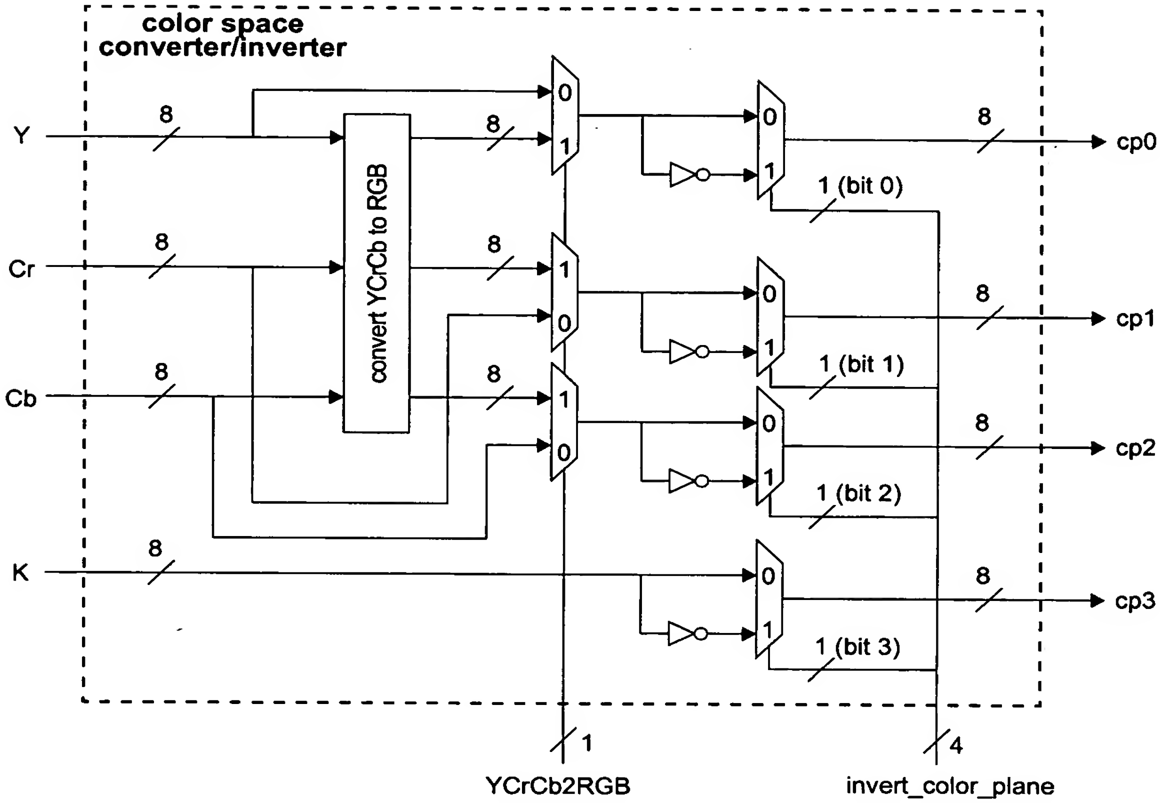


FIG. 145

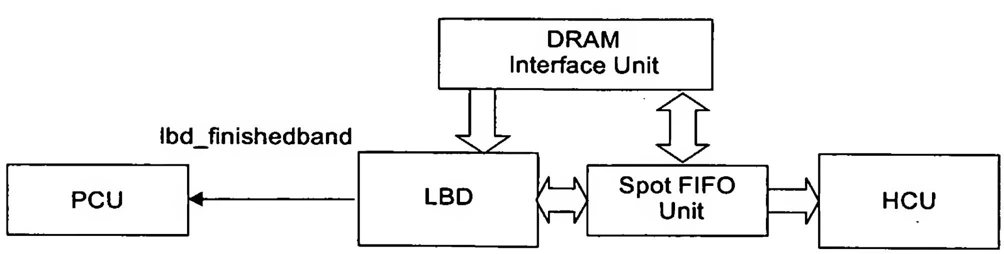


FIG. 146

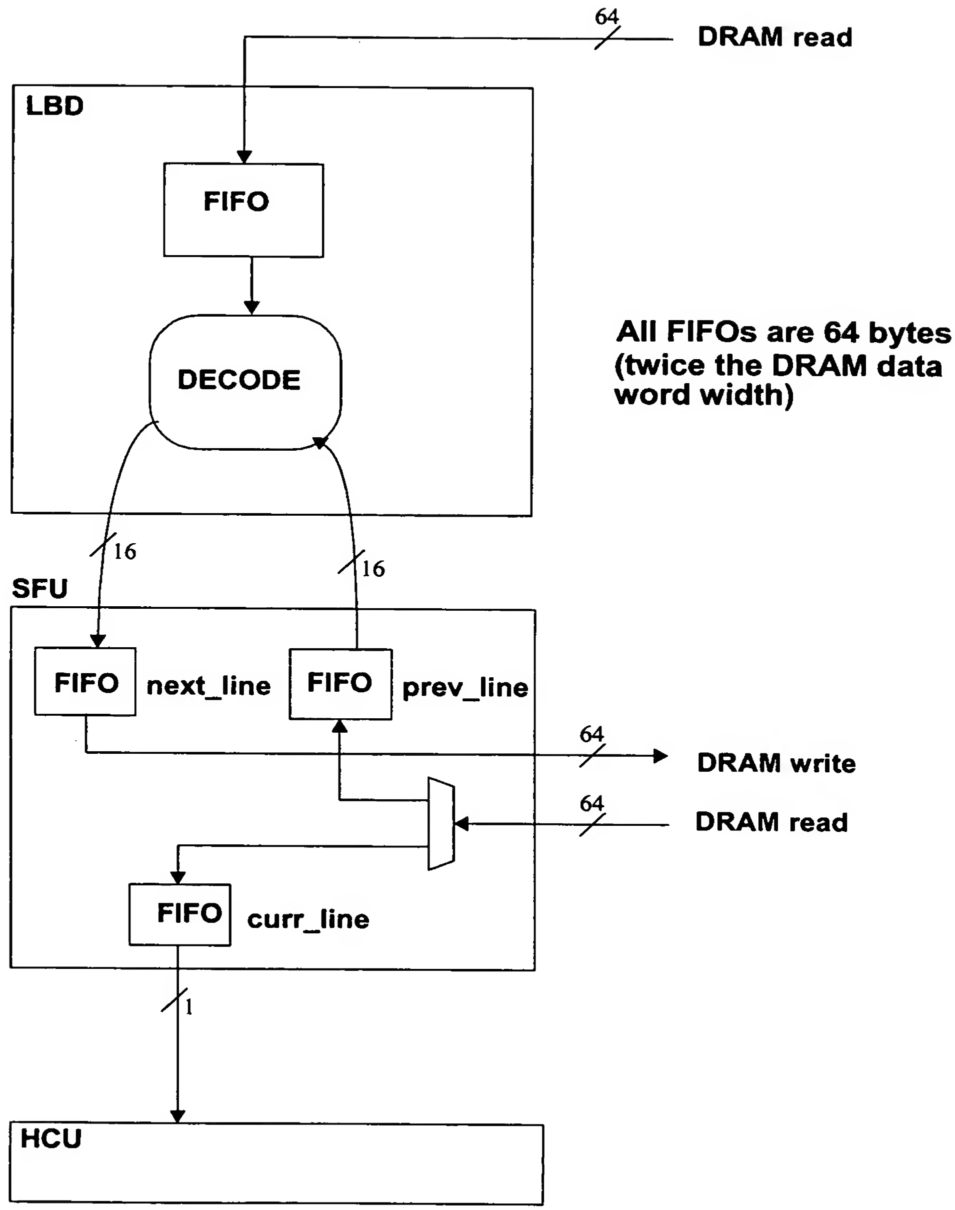
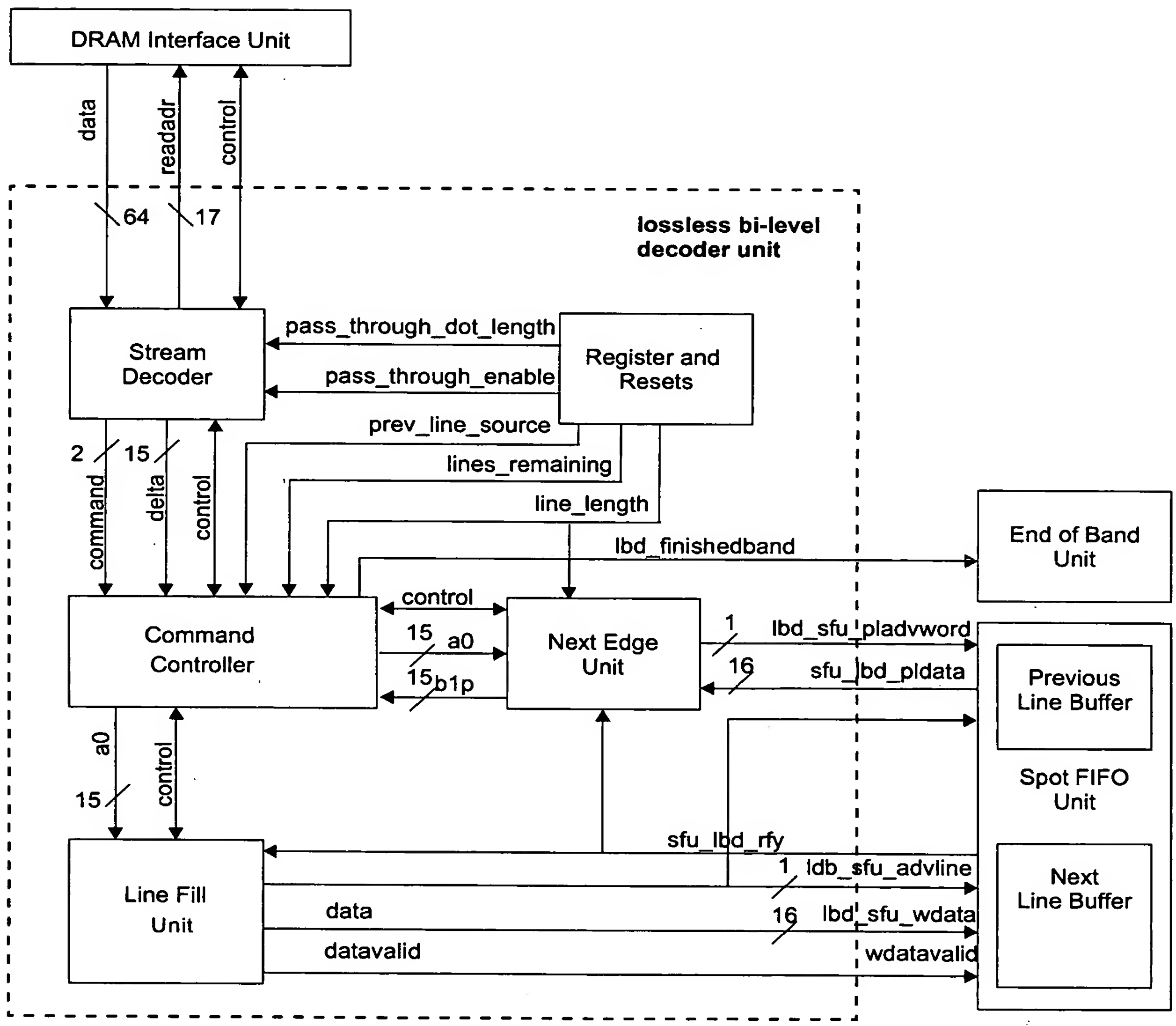

64
DRAM read
LBD
FIFO
DECODE
All FIFOs are 64 bytes (twice the DRAM data word width)
16
16
SFU
FIFO
next_line
FIFO
prev_line
64
DRAM write
64
DRAM read
FIFO
curr_line
1
HCU

FIG. 147



DRAM Interface Unit
data
readadr
control
64
17
lossless bi-level decoder unit
Stream Decoder
pass_through_dot_length
pass_through_enable
Register and Resets
prev_line_source
lines_remaining
line_length
2
15
command
delta
control
lbd_finishedband
End of Band Unit
control
Command Controller
15
a0
15
b1p
Next Edge Unit
1
lbd_sfu_pladvword
16
sfu_lbd_pldata
Previous Line Buffer
Spot FIFO Unit
a0
control
15
sfu_lbd_rfy
1
ldb_sfu_advline
Line Fill Unit
data
16
lbd_sfu_wdata
datavalid
wdatavalid
Next Line Buffer

FIG. 148

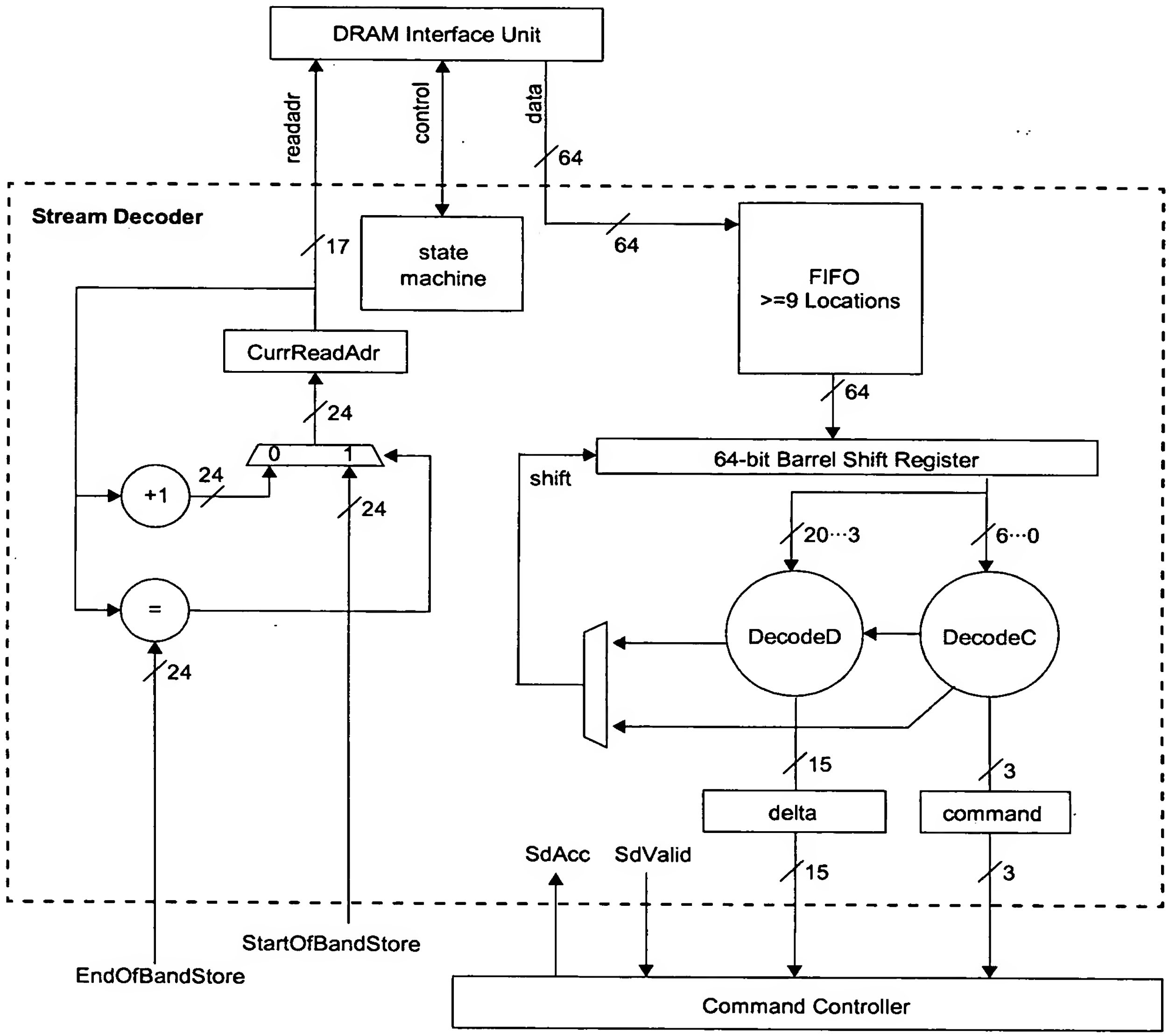

DRAM Interface Unit
readadr
control
data
64
Stream Decoder
17
state machine
64
FIFO >=9 Locations
CurrRead Adr
24
64
0
1
64-bit Barrel Shift Register
shift
+1
24
24
20···3
6···0
=
DecodeD
DecodeC
24
15
3
delta
command
SdAcc
SdValid
15
3
StartOfBandStore
EndOfBandStore
Command Controller

FIG. 149

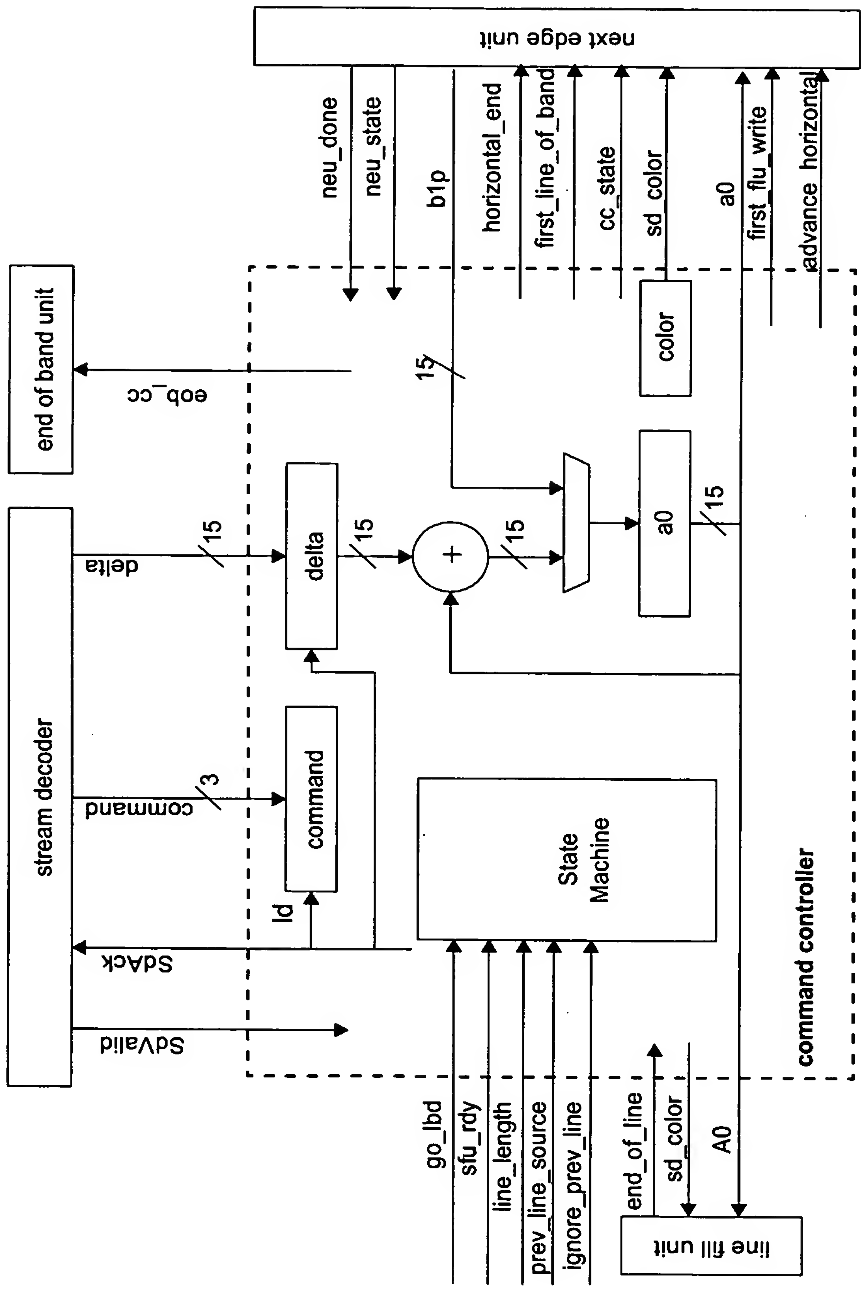

next edge unit
neu_done
neu_state
b1p
horizontal_end
first_line_of_band
cc_state
sd_color
a0
first_flu_write
advance_horizontal
end of band unit
eob_cc
15
color
stream decoder
delta
15
delta
15
+
15
a0
15
command
3
command
ld
State Machine
SdAck
SdValid
command controller
go_lbd
sfu_rdy
line_length
prev_line_source
ignore_prev_line
end_of_line
sd_color
A0
line fill unit

FIG. 150

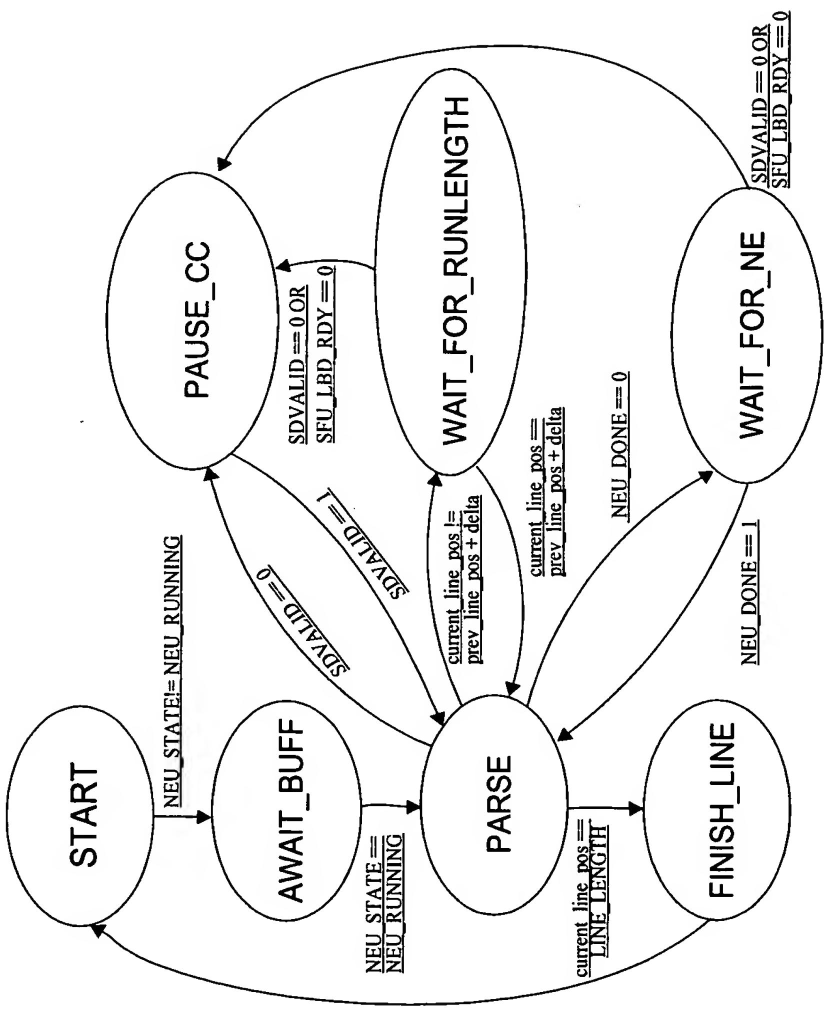

START
NEU_STATE!=NEU_RUNNING
AWAIT_BUFF
NEU_STATE == NEU_RUNNING
PARSE
current_line_pos == LINE_LENGTH
FINISH_LINE
SDVALID == 0
SDVALID == 1
PAUSE_CC
SDVALID == 0 OR SFU_LBD_RDY == 0
current_line_pos != prev_line_pos + delta
current_line_pos == prev_line_pos + delta
WAIT_FOR_RUNLENGTH
NEU_DONE == 0
NEU_DONE == 1
WAIT_FOR_NE
SDVALID == 0 OR SFU_LBD_RDY == 0

FIG. 151

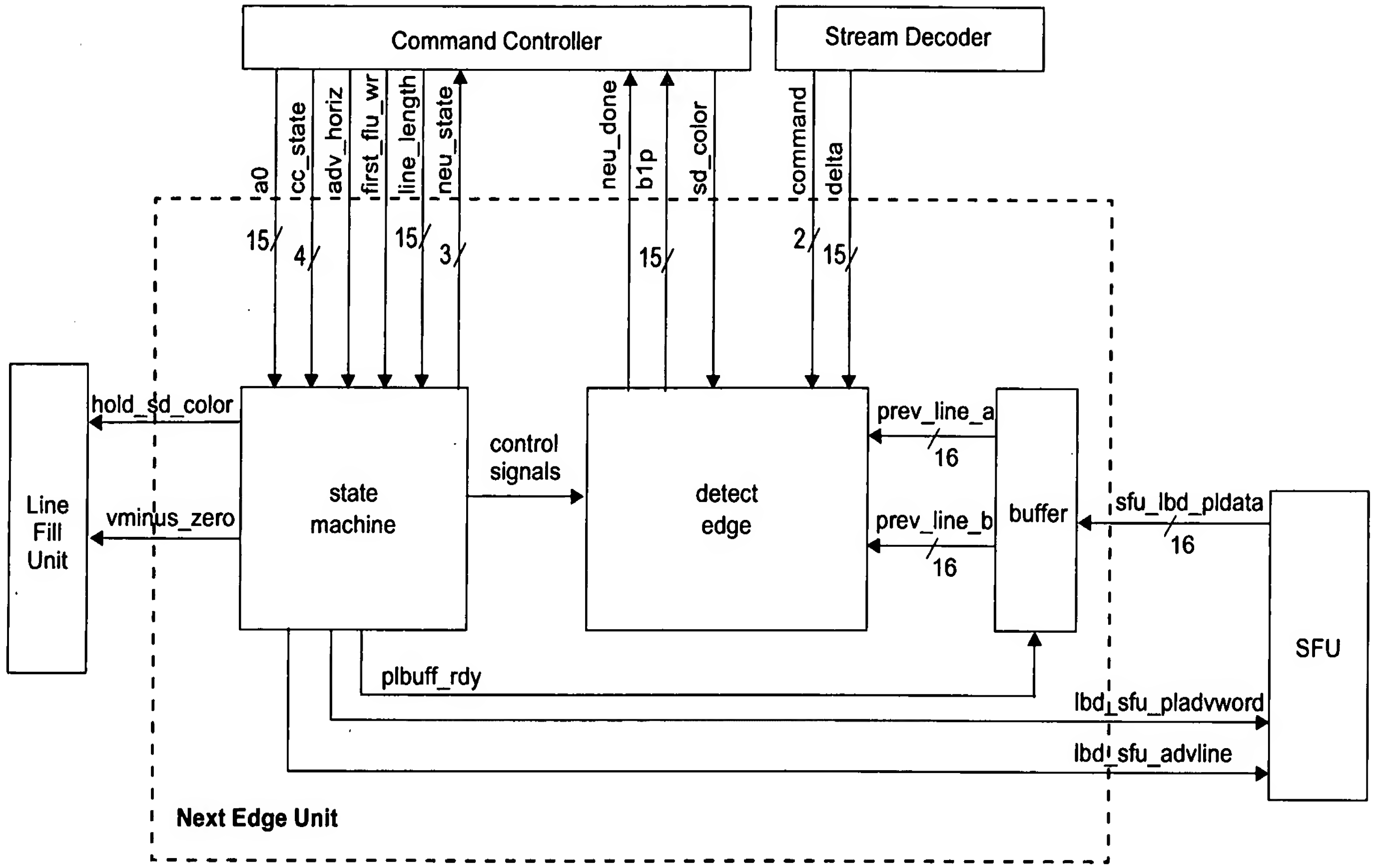

Command Controller
Stream Decoder
a0
15
cc_state
4
adv_horiz
first_flu_wr
line_length
15
neu_state
3
neu_done
b1p
15
sd_color
command
2
delta
15
Line
Fill
Unit
hold_sd_color
vminus_zero
state
machine
control
signals
detect
edge
prev_line_a
16
prev_line_b
16
buffer
sfu_lbd_pldata
16
SFU
plbuff_rdy
lbd_sfu_pladvword
lbd_sfu_advline
Next Edge Unit

FIG. 152

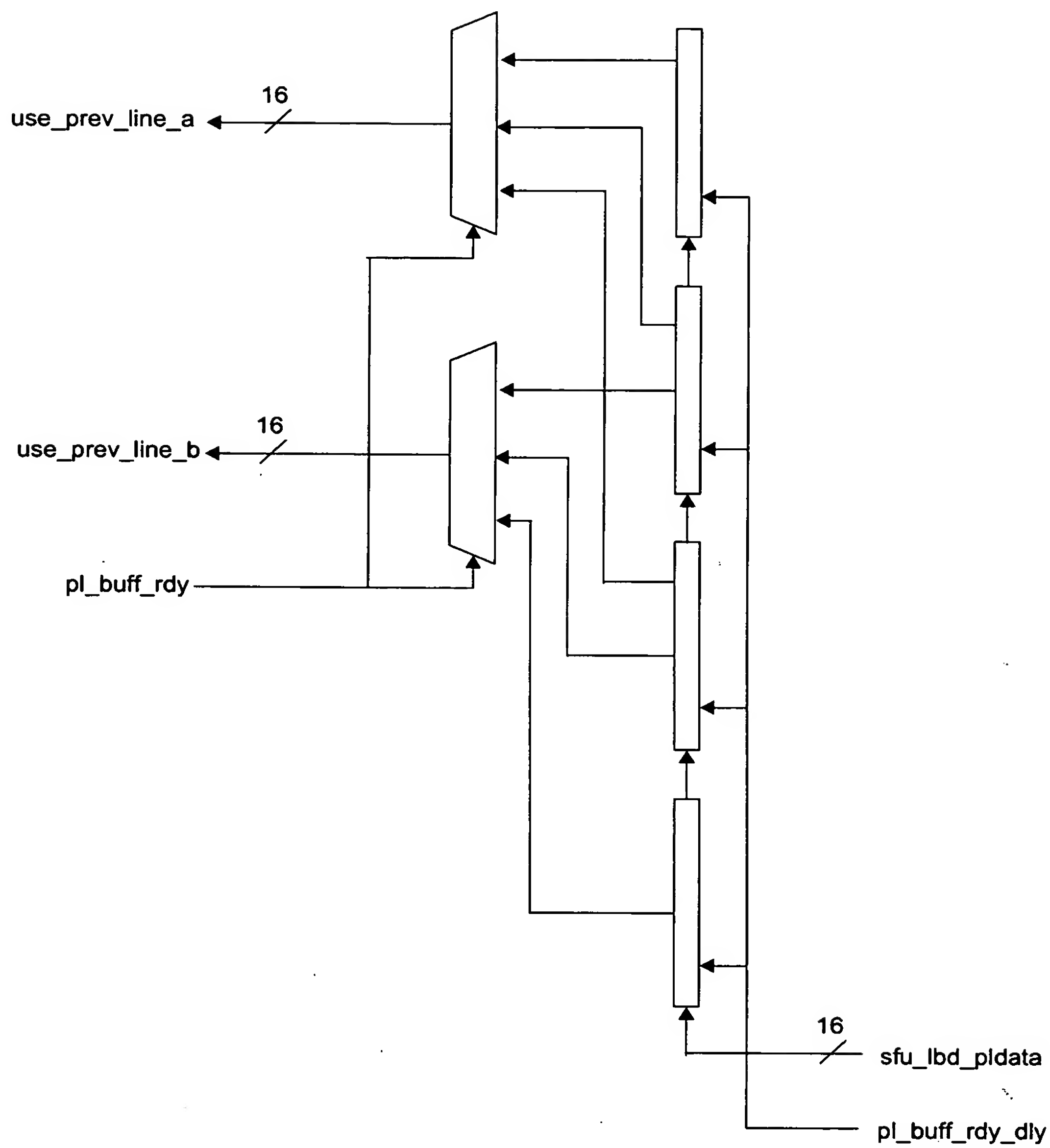

16
use_prev_line_a
16
use_prev_line_b
pl_buff_rdy
16
sfu_lbd_pldata
pl_buff_rdy_dly

FIG. 153

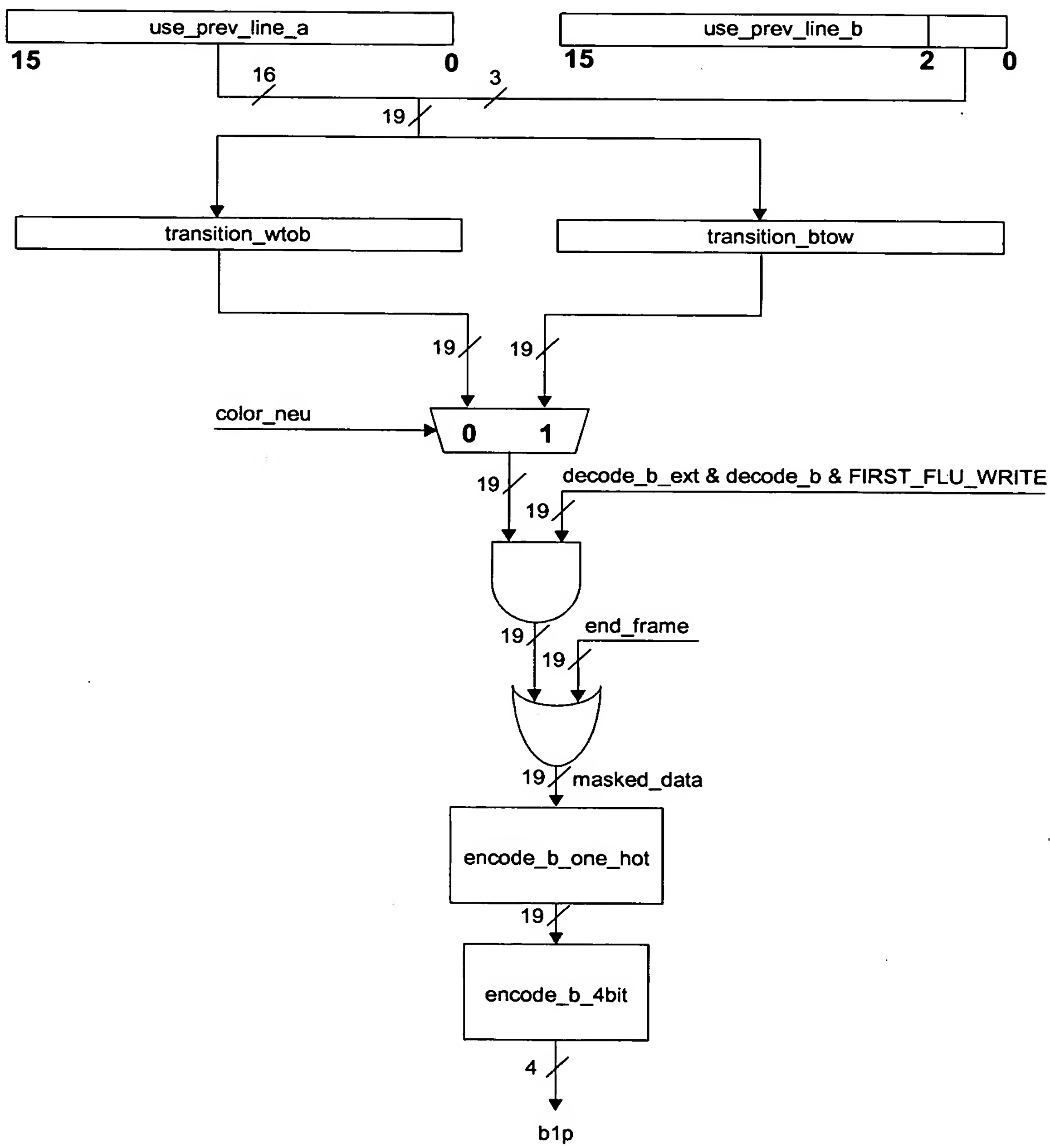

use_prev_line_a
15
0
use_prev_line_b
15
2
0
16
3
19
transition_wtob
transition_btow
19
19
color_neu
0
1
19
decode_b_ext & decode_b & FIRST_FLU_WRITE
19
19
end_frame
19
19
masked_data
encode_b_one_hot
19
encode_b_4bit
4
b1p

FIG. 154

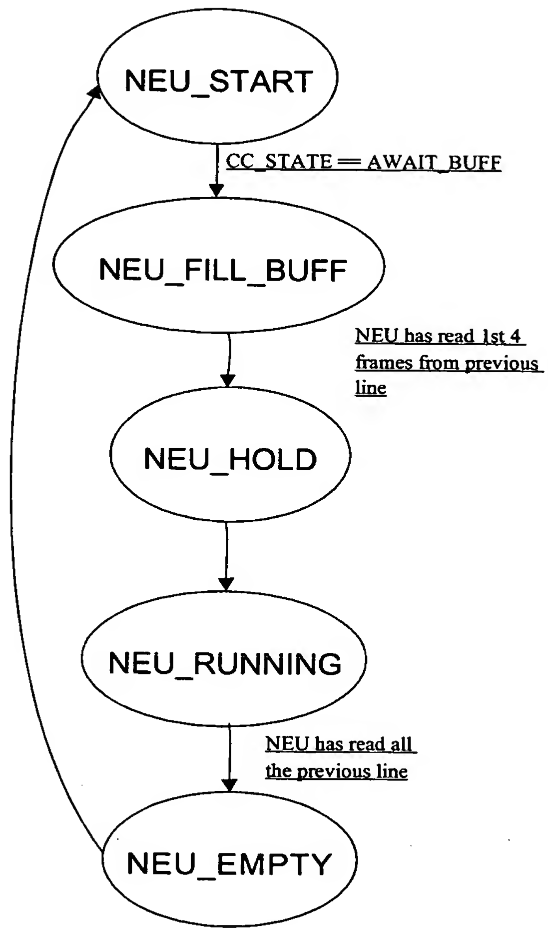

NEU_START
CC_STATE == AWAIT_BUFF
NEU_FILL_BUFF
NEU has read 1st 4 frames from previous line
NEU_HOLD
NEU_RUNNING
NEU has read all the previous line
NEU_EMPTY

FIG. 155

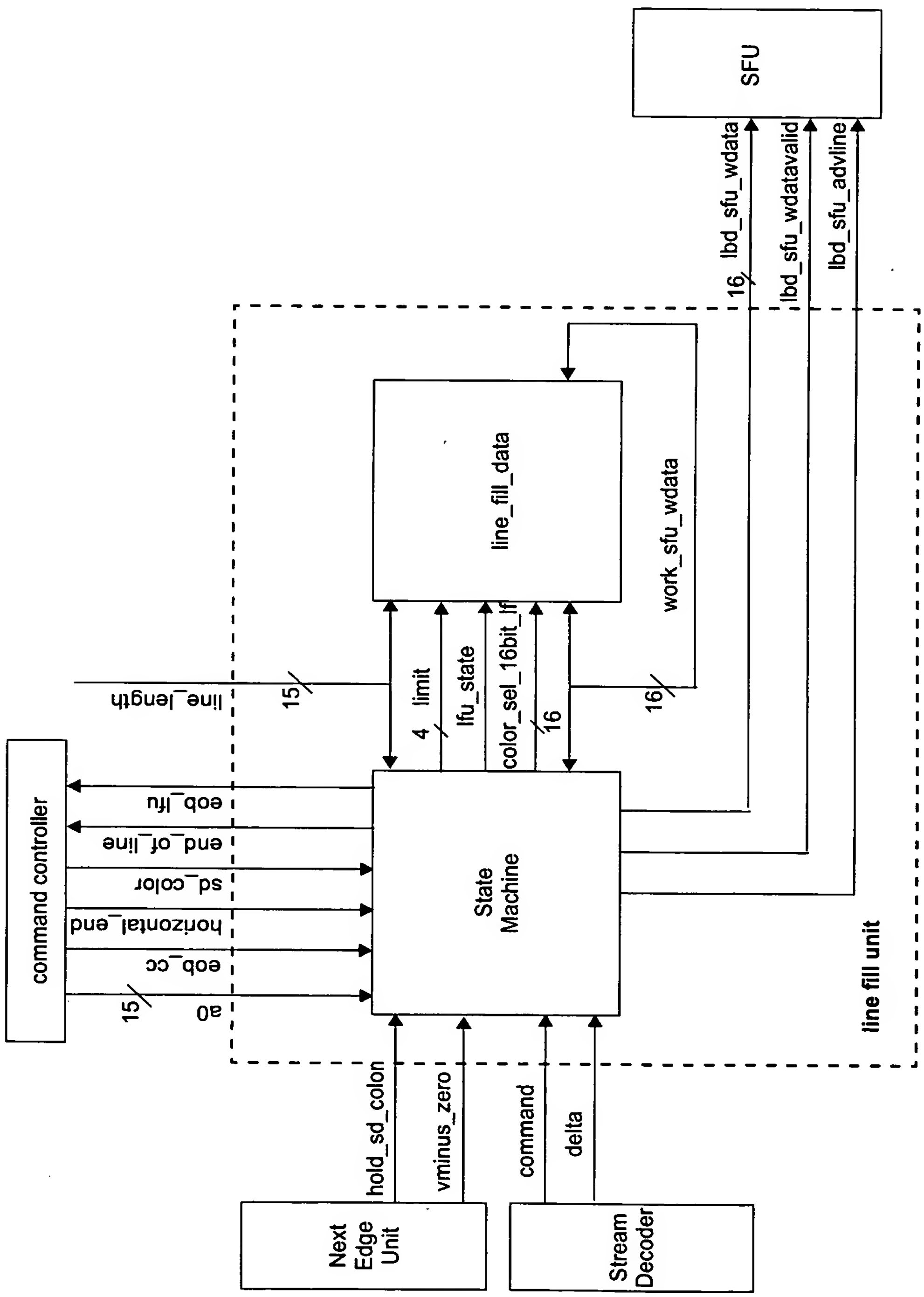

SFU
lbd_sfu_wdata
16
lbd_sfu_wdatavalid
lbd_sfu_adviine
line_fill_data
work_sfu_wdata
16
line_length
15
4
limit
lfu_state
color_sel_16bit_lf
16
command controller
eob_lfu
end_of_line
sd_color
horizontal_end
eob_cc
15
a0
State
Machine
line fill unit
hold_sd_colon
vminus_zero
command
delta
Next
Edge
Unit
Stream
Decoder

FIG. 156

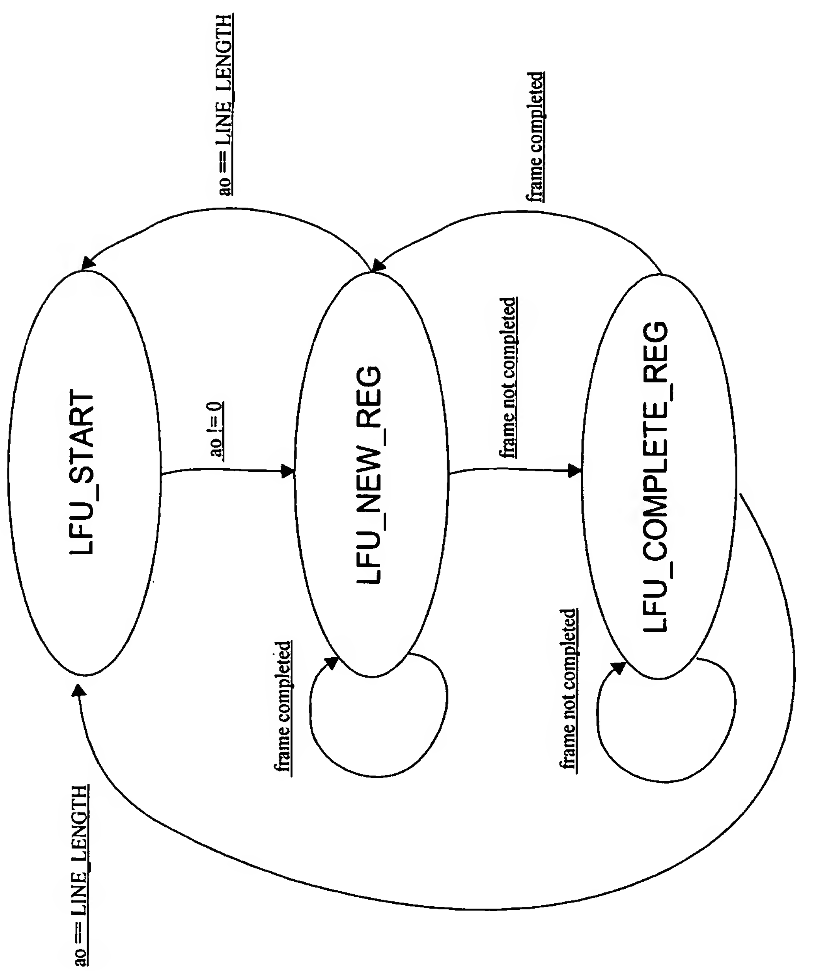

LFU_START
LFU_NEW_REG
LFU_COMPLETE_REG
ao != 0
frame not completed
frame completed
frame not completed
frame completed
ao == LINE_LENGTH
ao == LINE_LENGTH

FIG. 157

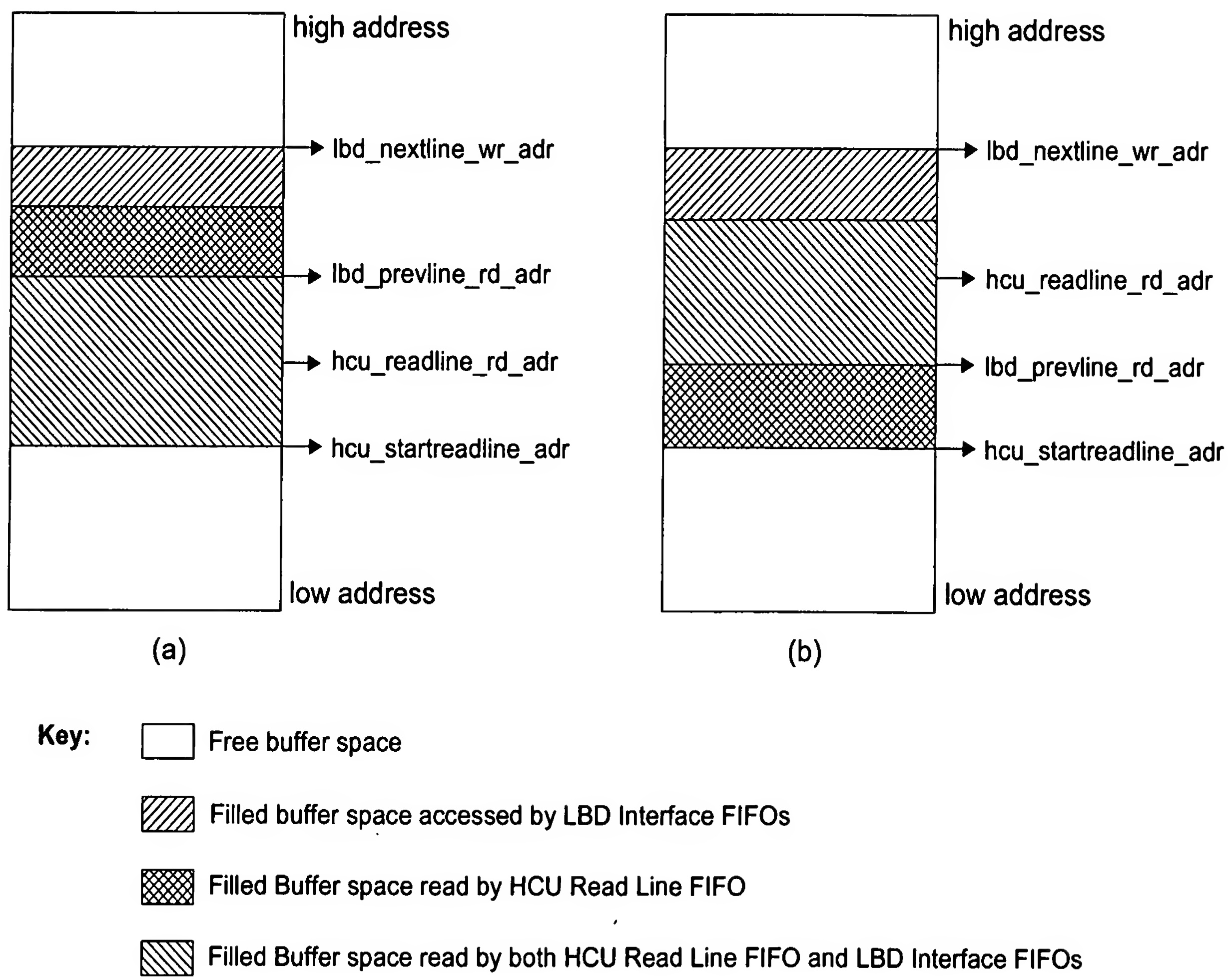

high address
lbd_nextline_wr_adr
lbd_prevline_rd_adr
hcu_readline_rd_adr
hcu_startreadline_adr
low address
(a)
high address
lbd_nextline_wr_adr
hcu_readline_rd_adr
lbd_prevline_rd_adr
hcu_startreadline_adr
low address
(b)
Key:
Free buffer space
Filled buffer space accessed by LBD Interface FIFOs
Filled Buffer space read by HCU Read Line FIFO
Filled Buffer space read by both HCU Read Line FIFO and LBD Interface FIFOs

FIG. 158

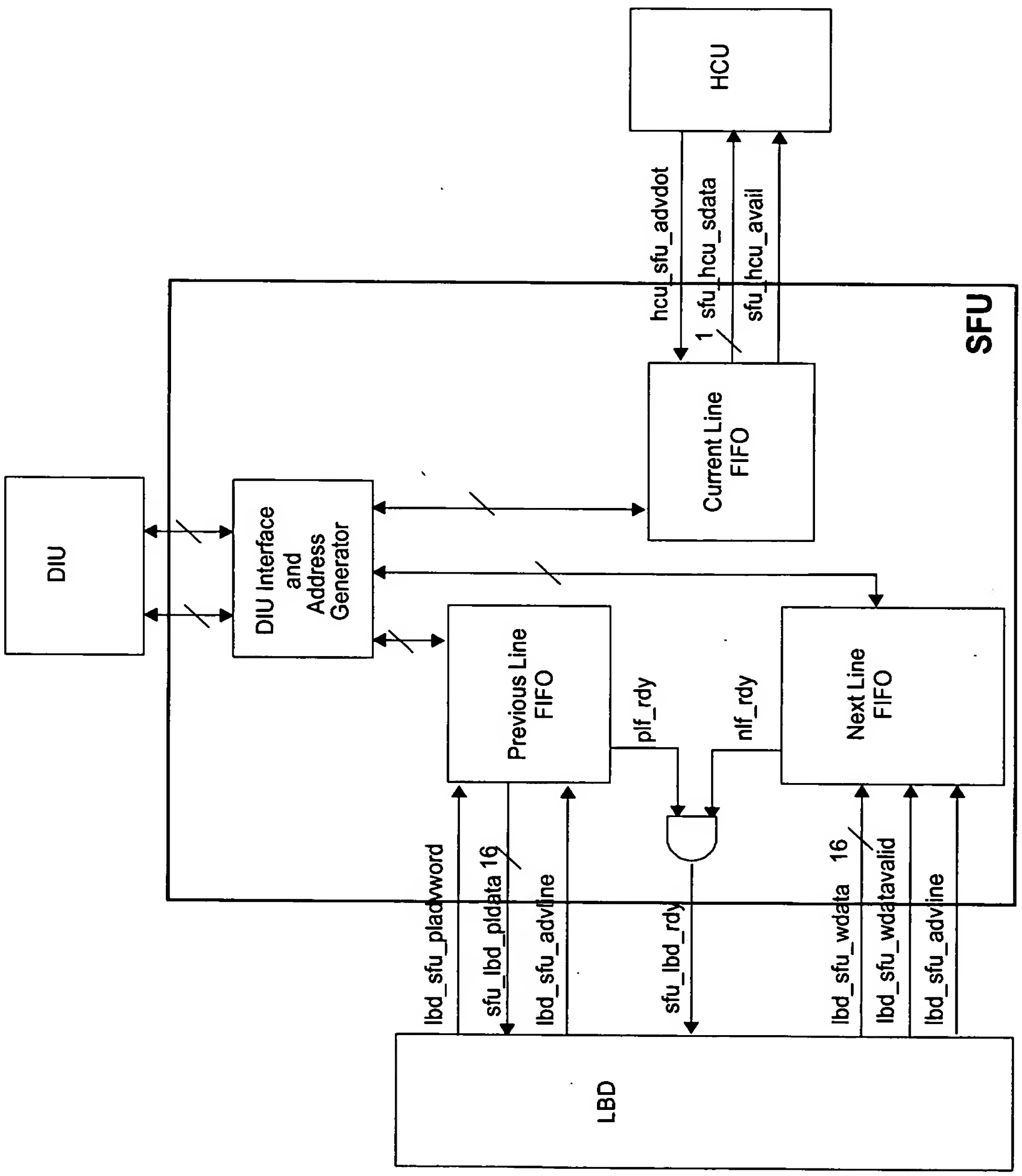

HCU
hcu_sfu_advdot
1
sfu_hcu_sdata
sfu_hcu_avail
SFU
DIU
DIU Interface and Address Generator
Current Line FIFO
Previous Line FIFO
Next Line FIFO
plf_rdy
nlf_rdy
lbd_sfu_pladvword
sfu_lbd_pldata 16
lbd_sfu_advline
sfu_lbd_rdy
lbd_sfu_wdata 16
lbd_sfu_wdatavalid
lbd_sfu_advline
LBD

FIG. 159

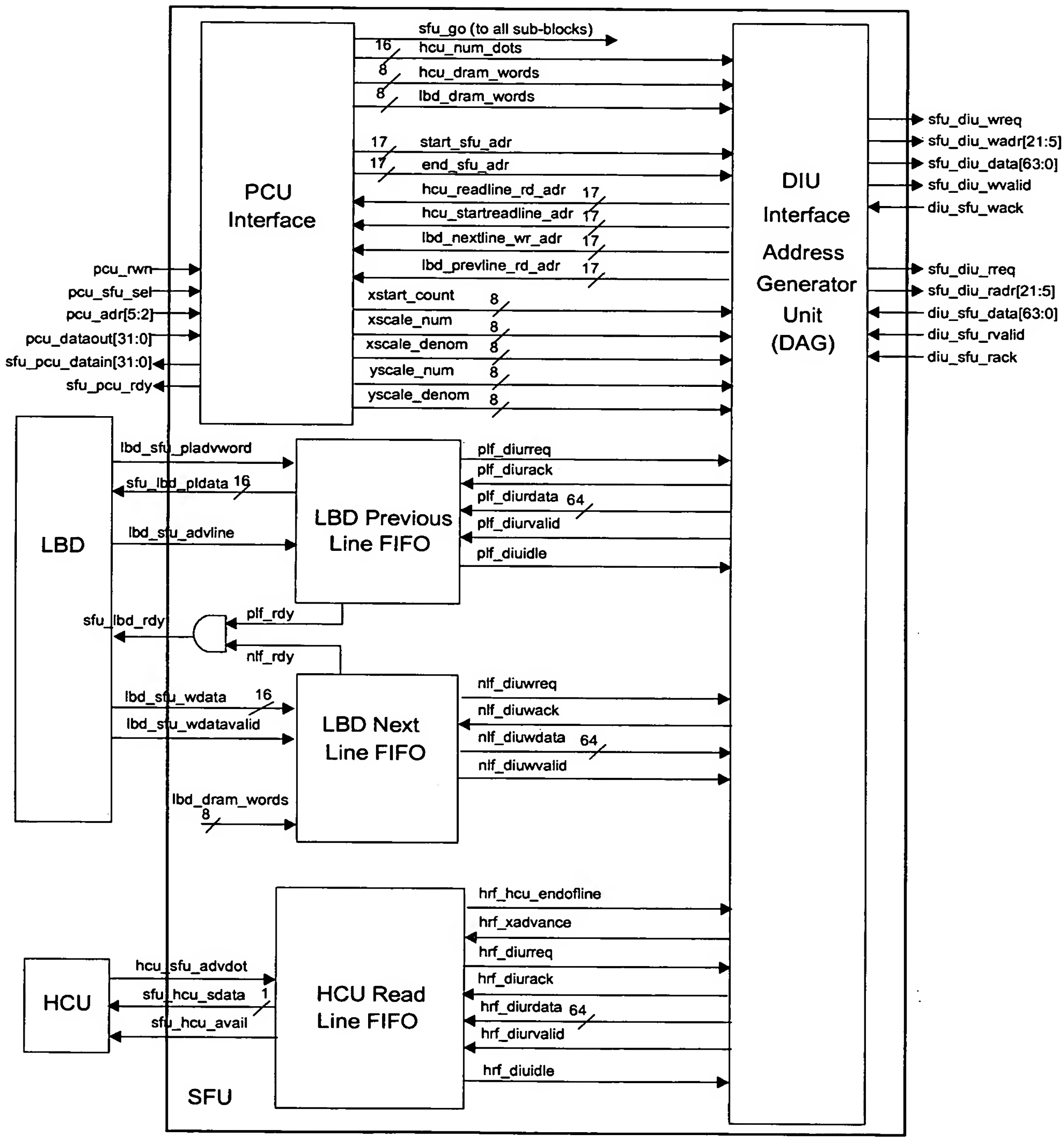

PCU Interface
sfu_go (to all sub-blocks)
hcu_num_dots
16
hcu_dram_words
8
lbd_dram_words
8
start_sfu_adr
17
end_sfu_adr
17
hcu_readline_rd_adr
17
hcu_startreadline_adr
17
lbd_nextline_wr_adr
17
lbd_prevline_rd_adr
17
xstart_count
8
xscale_num
8
xscale_denom
8
yscale_num
8
yscale_denom
8
pcu_rwn
pcu_sfu_sel
pcu_adr[5:2]
pcu_dataout[31:0]
sfu_pcu_datain[31:0]
sfu_pcu_rdy
DIU Interface Address Generator Unit (DAG)
sfu_diu_wreq
sfu_diu_wadr[21:5]
sfu_diu_data[63:0]
sfu_diu_wvalid
diu_sfu_wack
sfu_diu_rreq
sfu_diu_radr[21:5]
diu_sfu_data[63:0]
diu_sfu_rvalid
diu_sfu_rack
LBD
lbd_sfu_pladvword
sfu_lbd_pldata
16
lbd_sfu_advline
LBD Previous Line FIFO
plf_diurreq
plf_diurack
plf_diurdata
64
plf_diurvalid
plf_diuidle
sfu_lbd_rdy
plf_rdy
nlf_rdy
lbd_sfu_wdata
16
lbd_sfu_wdatavalid
LBD Next Line FIFO
nlf_diuwreq
nlf_diuwack
nlf_diuwdata
64
nlf_diuwvalid
lbd_dram_words
8
HCU Read Line FIFO
hrf_hcu_endofline
hrf_xadvance
hrf_diurreq
hrf_diurack
hrf_diurdata
64
hrf_diurvalid
hrf_diuidle
HCU
hcu_sfu_advdot
sfu_hcu_sdata
1
sfu_hcu_avail
SFU

FIG. 160

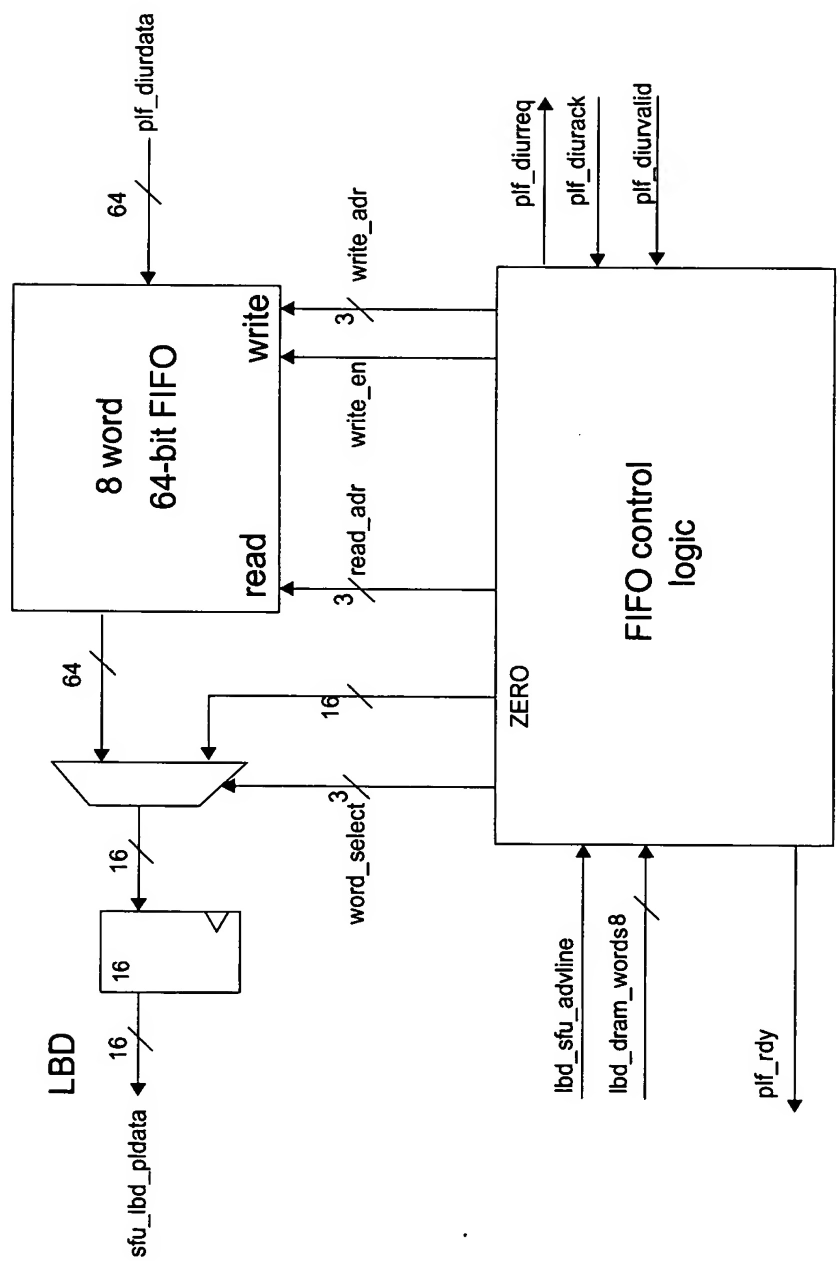

LBD
8 word
64-bit FIFO
read
write
FIFO control
logic
ZERO
plf_diurdata
64
write_adr
3
write_en
read_adr
3
64
16
16
word_select
3
16
16
16
plf_diurreq
plf_diurack
plf_diurvalid
lbd_sfu_advline
lbd_dram_words8
plf_rdy
sfu_lbd_pldata

FIG. 161

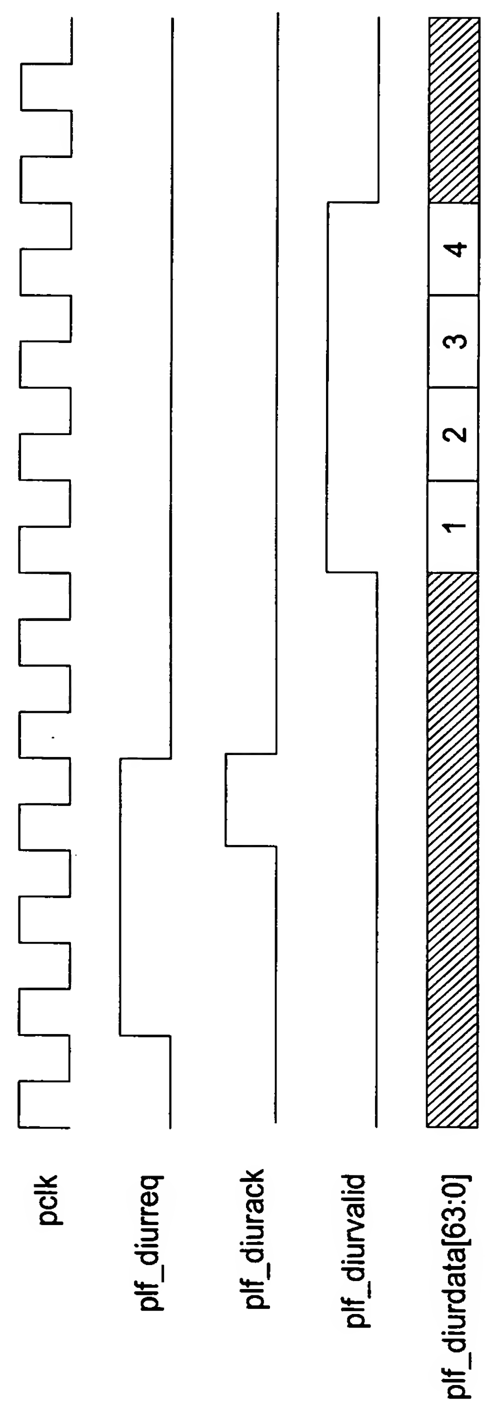

pclk
plf_diurreq
plf_diurack
plf_diurvalid
plf_diurdata[63:0]
1
2
3
4

FIG. 162

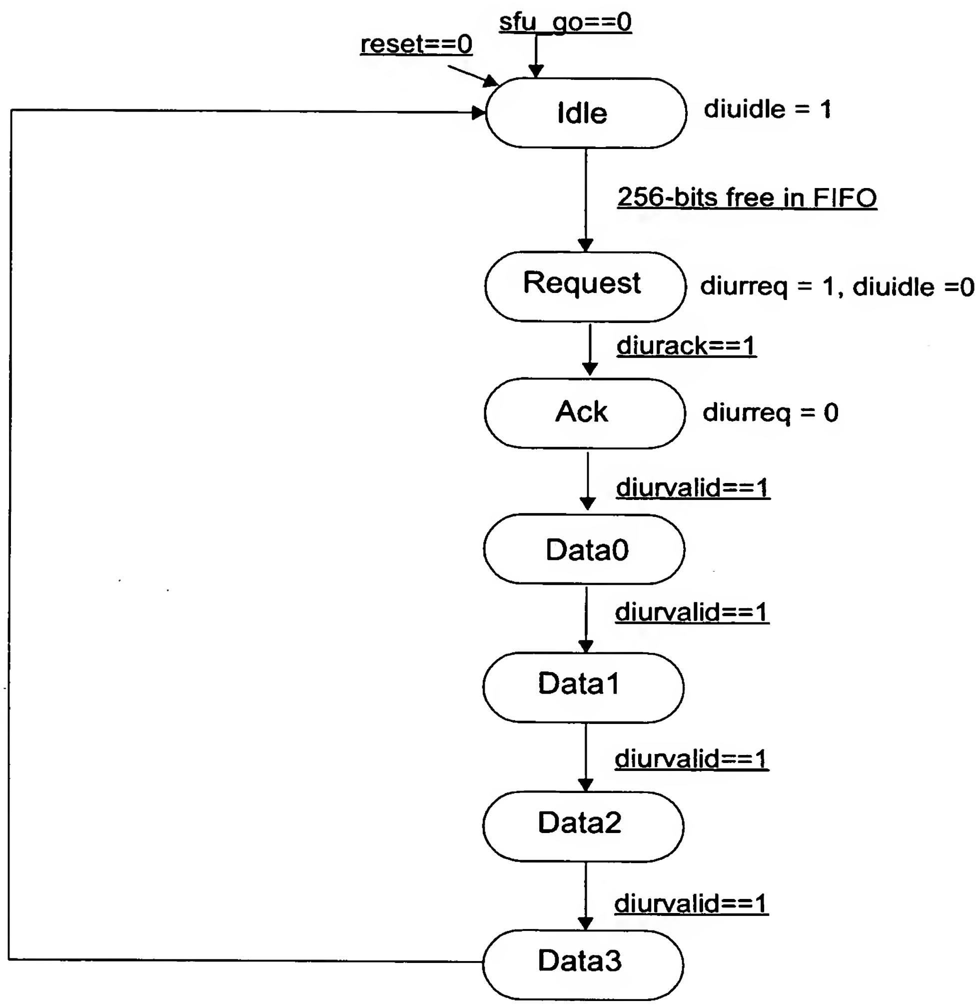

sfu_go==0
reset==0
Idle
diuidle = 1
256-bits free in FIFO
Request
diurreq = 1, diuidle =0
diurack==1
Ack
diurreq = 0
diurvalid==1
Data0
diurvalid==1
Data1
diurvalid==1
Data2
diurvalid==1
Data3

FIG. 163

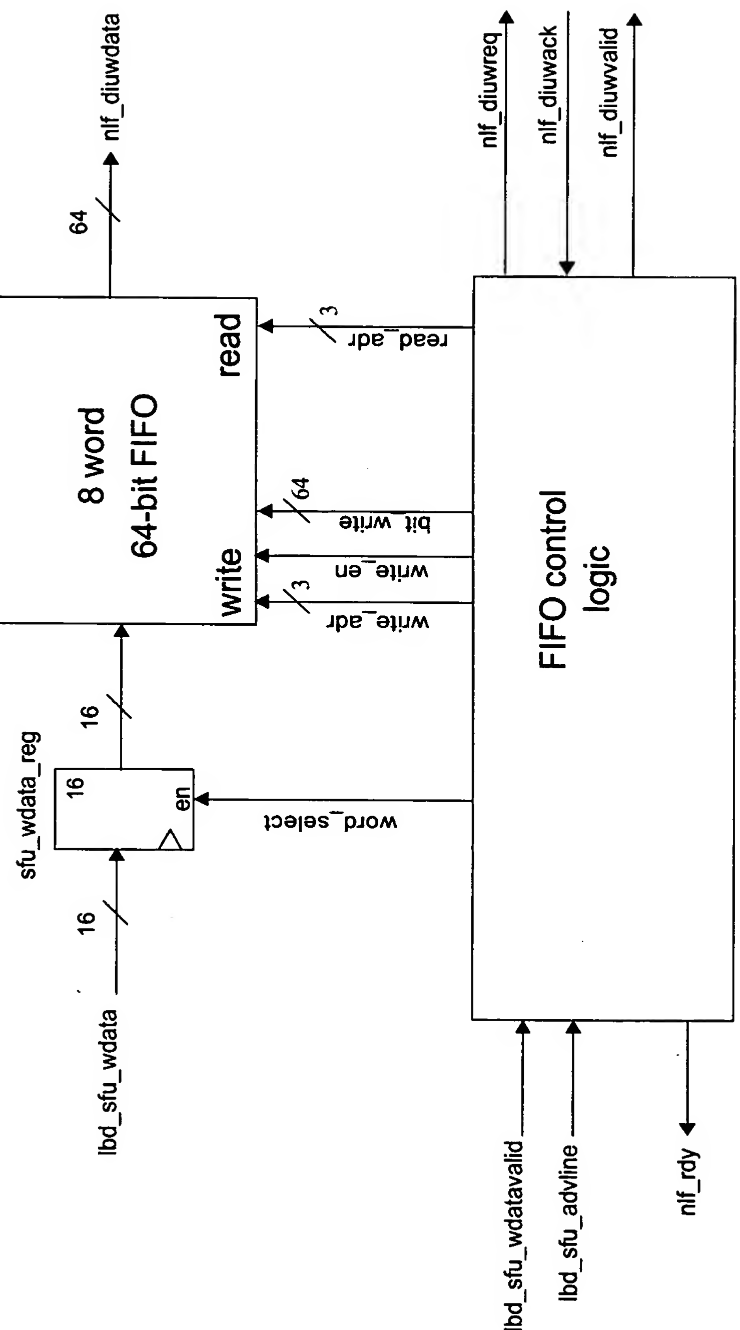

nlf_diuwdata
64
8 word
64-bit FIFO
read
write
read_adr
3
bit_write
64
write_en
write_adr
3
16
sfu_wdata_reg
16
en
word_select
16
lbd_sfu_wdata
FIFO control
logic
nlf_diuwreq
nlf_diuwack
nlf_diuwvalid
lbd_sfu_wdatavalid
lbd_sfu_advline
nlf_rdy

FIG. 164

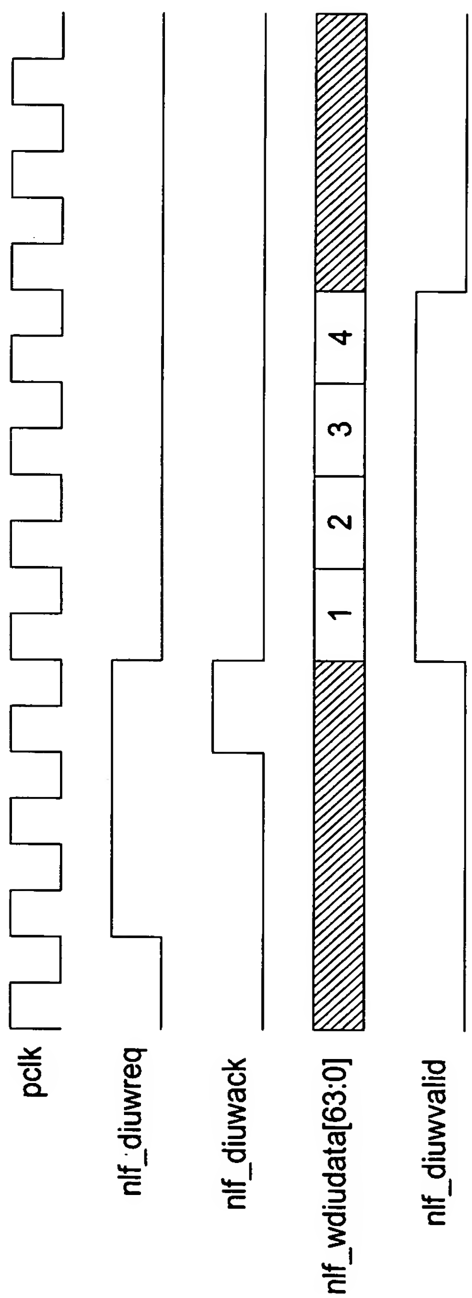

pclk
nlf_diuwreq
nlf_diuwack
nlf_wdiudata[63:0]
nlf_diuwvalid
1
2
3
4

FIG. 165

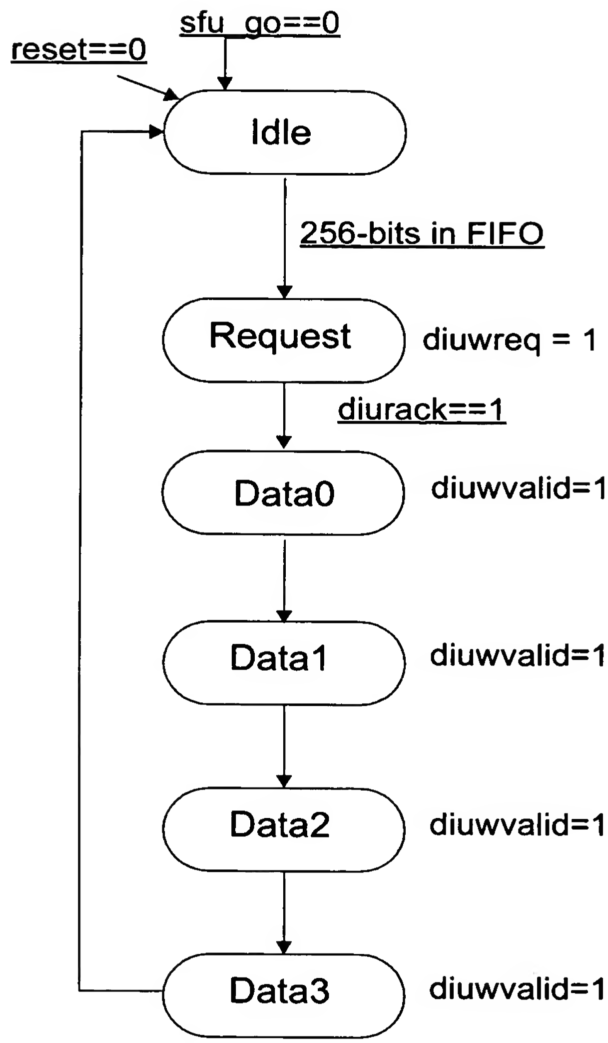

sfu_go==0
reset==0
Idle
256-bits in FIFO
Request
diuwreq = 1
diurack==1
Data0
diuwvalid=1
Data1
diuwvalid=1
Data2
diuwvalid=1
Data3
diuwvalid=1

FIG. 166

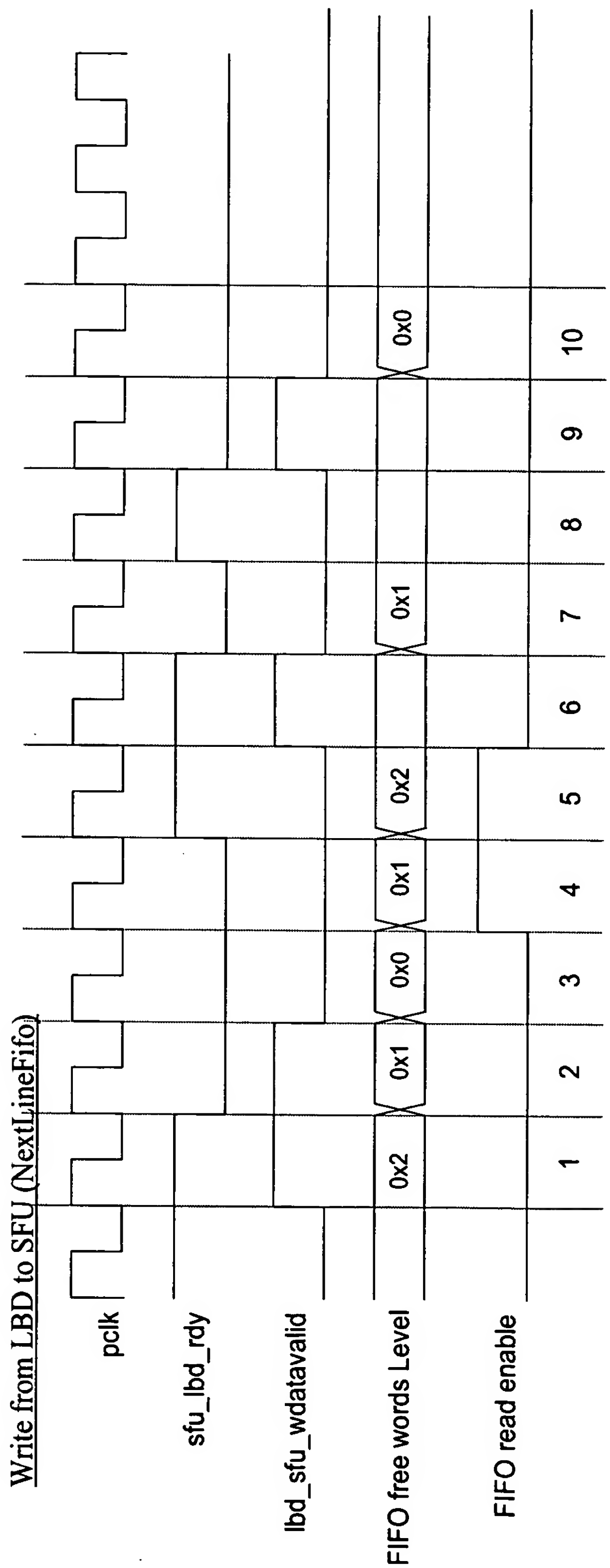

Write from LBD to SFU (NextLineFifo)
pclk
sfu_lbd_rdy
lbd_sfu_wdatavalid
FIFO free words Level
FIFO read enable
0x2
0x1
0x0
0x1
0x2
0x1
0x0
1
2
3
4
5
6
7
8
9
10

FIG. 167

Read from SFU to LBD (PreviousLineFifo)

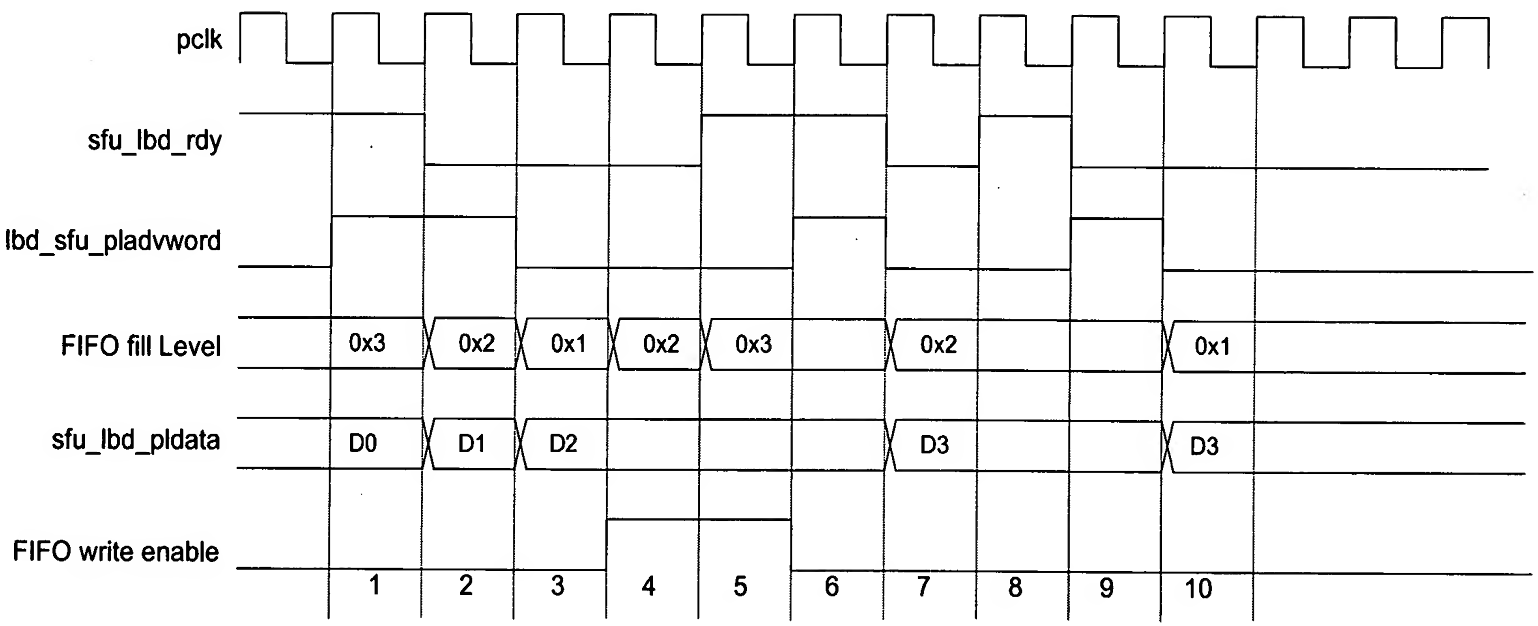


FIG. 168

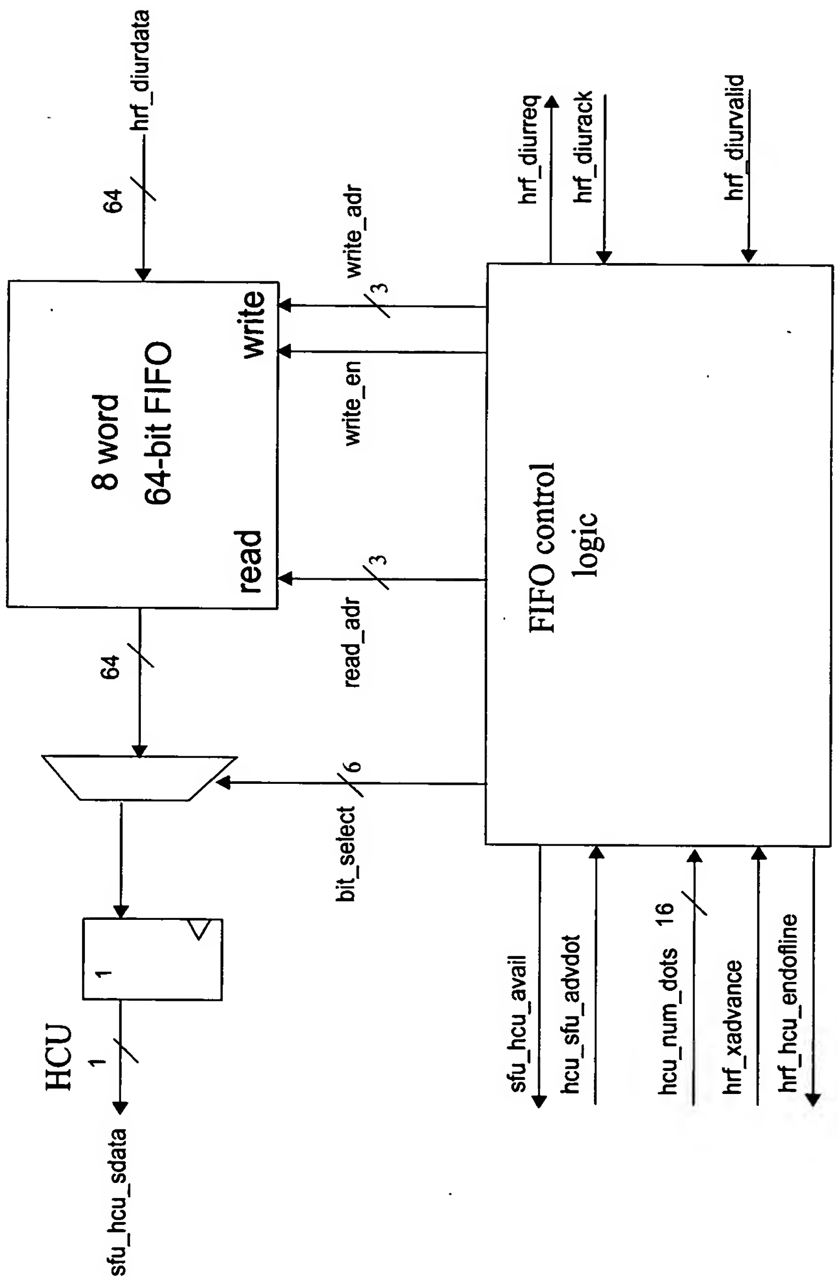

hrf_diurdata
64
8 word
64-bit FIFO
write
read
write_adr
3
write_en
read_adr
3
64
bit_select
6
FIFO control
logic
hrf_diurreq
hrf_diurack
hrf_diurvalid
1
HCU
1
sfu_hcu_sdata
sfu_hcu_avail
hcu_sfu_advdot
hcu_num_dots
16
hrf_xadvance
hrf_hcu_endofline

FIG. 169

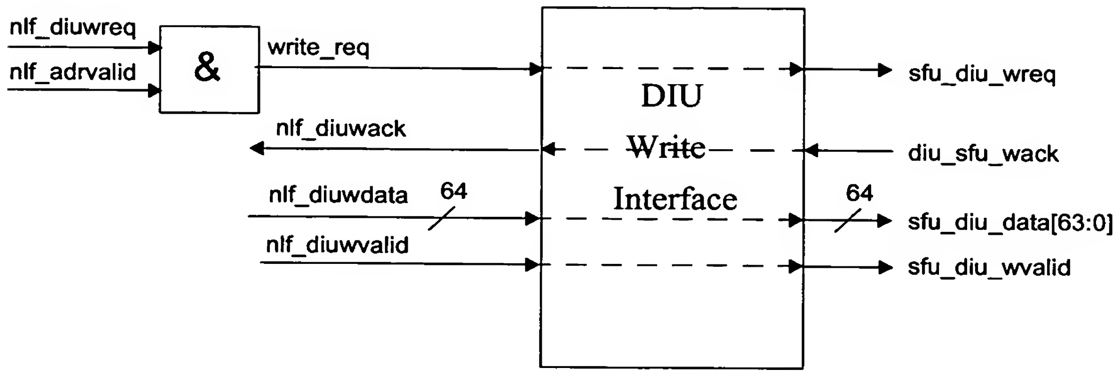


FIG. 170

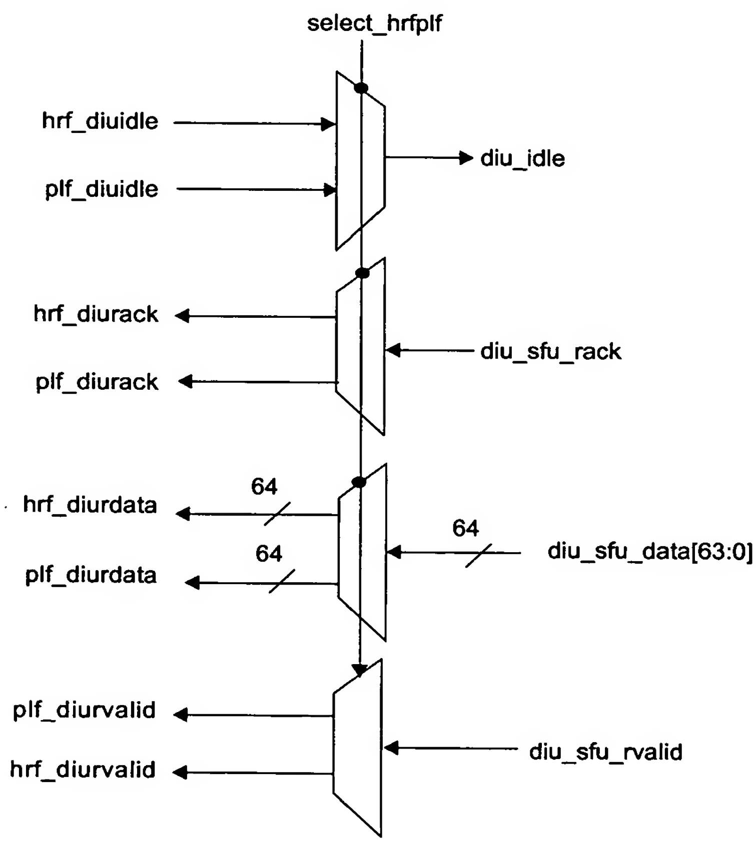


FIG. 171

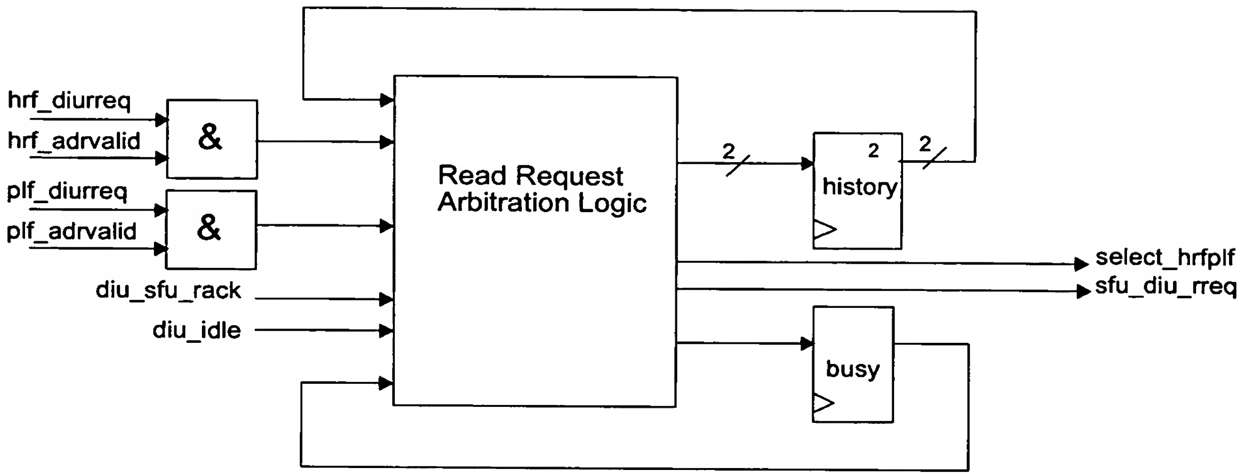


FIG. 172

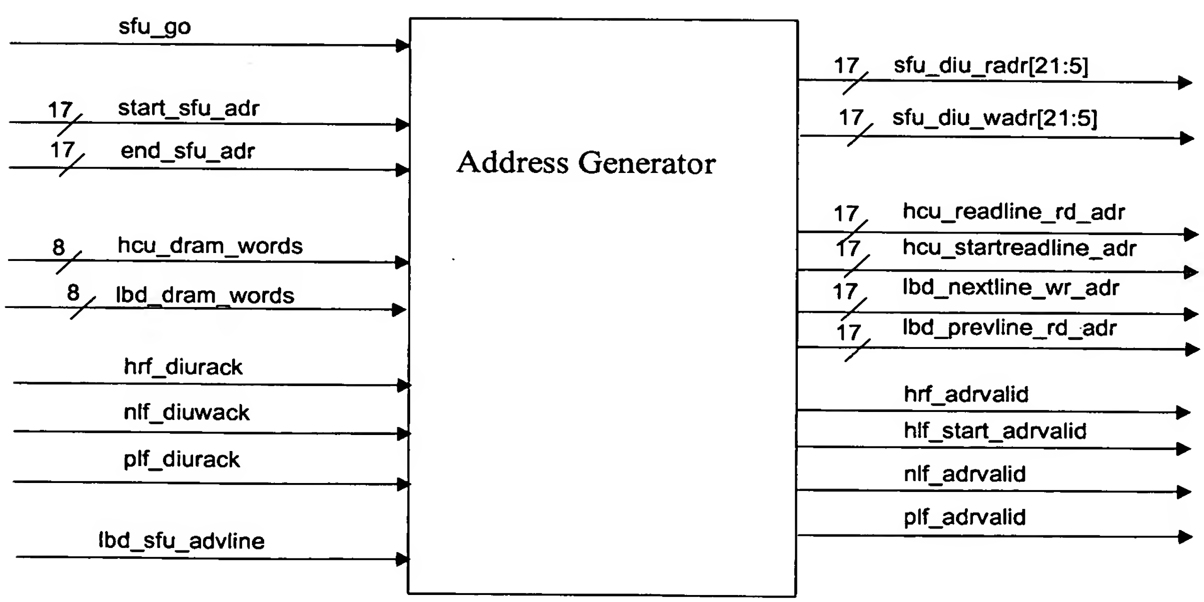


FIG. 173

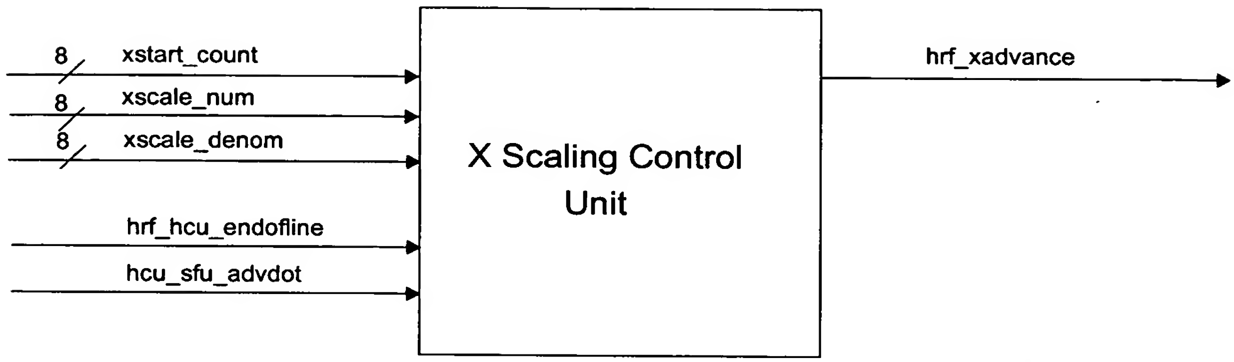


FIG. 174

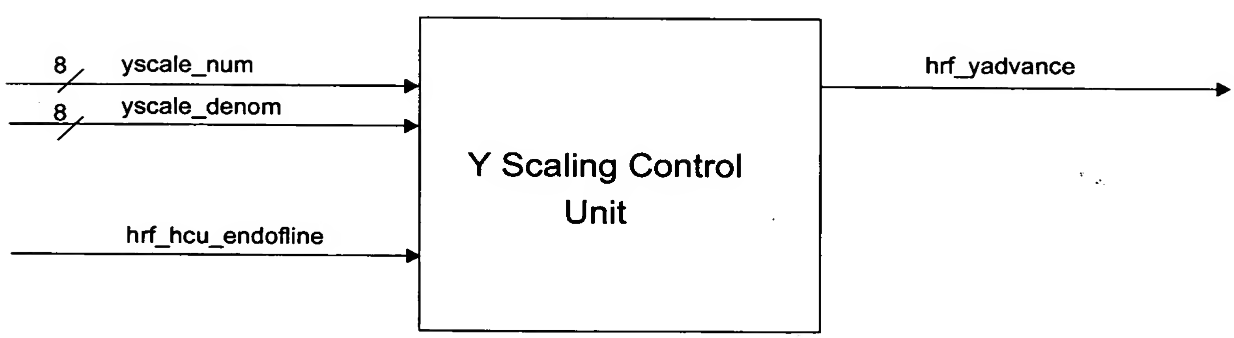


FIG. 175

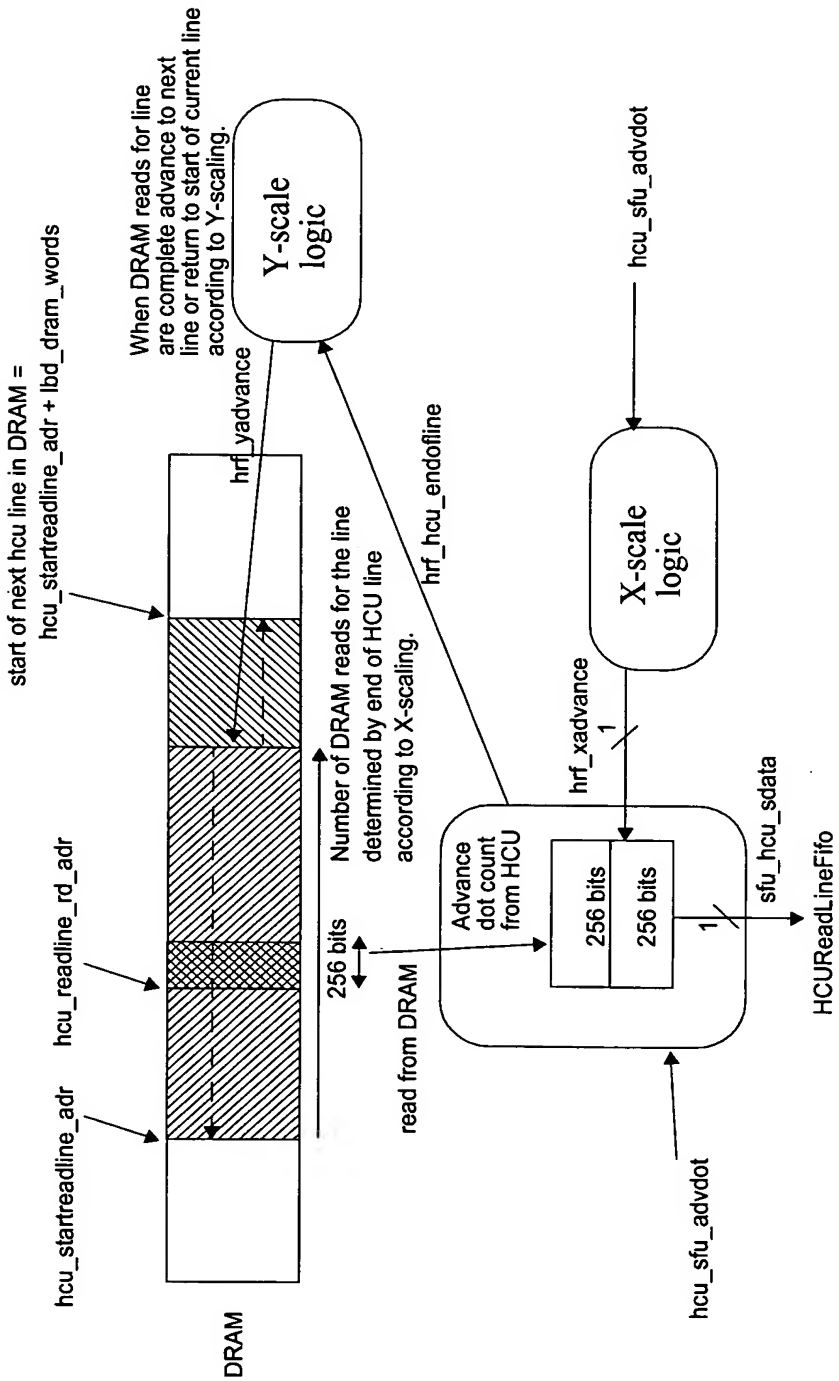

start of next hcu line in DRAM =
hcu_startreadline_adr + lbd_dram_words
hcu_readline_rd_adr
hcu_startreadline_adr
DRAM
hrf_yadvance
When DRAM reads for line are complete advance to next line or return to start of current line according to Y-scaling.
Y-scale logic
256 bits
read from DRAM
Number of DRAM reads for the line determined by end of HCU line according to X-scaling.
hrf_hcu_endofline
hcu_sfu_advdot
X-scale logic
hrf_xadvance
1
Advance dot count from HCU
256 bits
256 bits
1
sfu_hcu_sdata
HCUReadLineFifo
hcu_sfu_advdot

FIG. 176

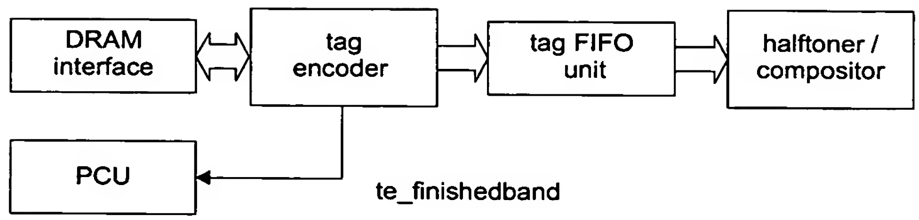


FIG. 177

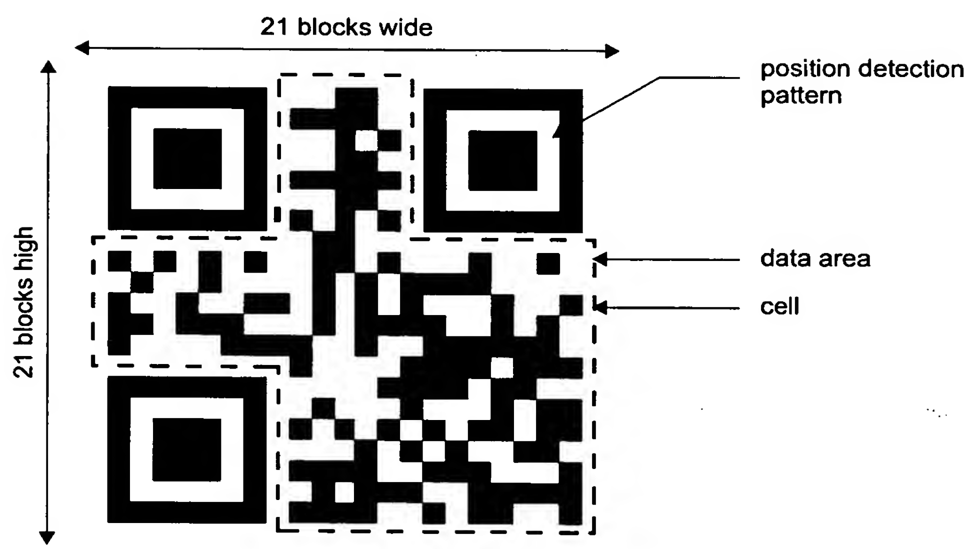
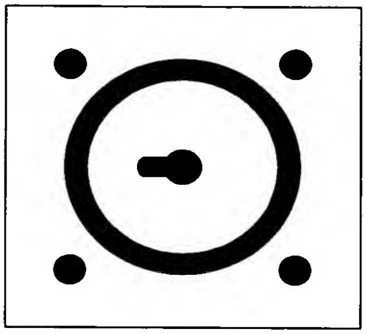
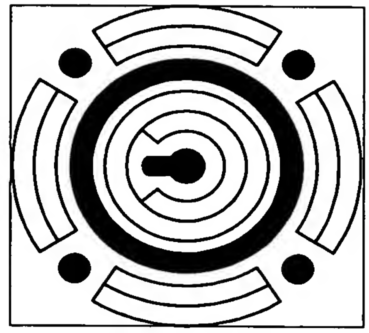


FIG. 178



(a) Netpage tag background pattern



(b) Netpage tag showing data area

FIG. 179

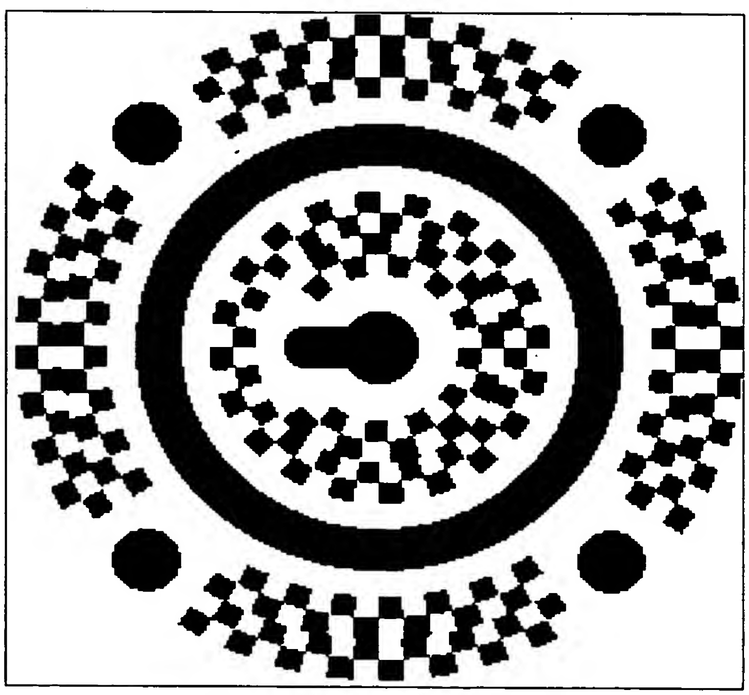


FIG. 180

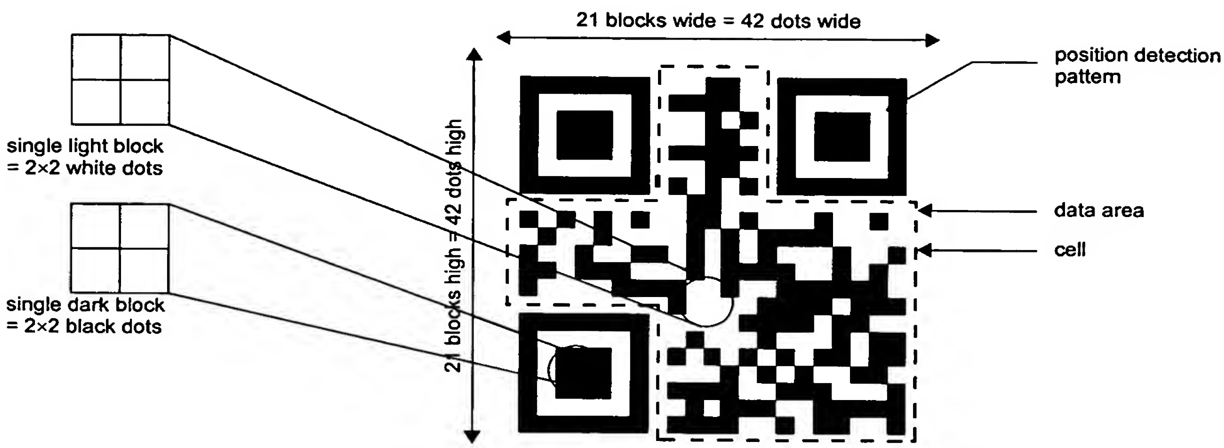


FIG. 181

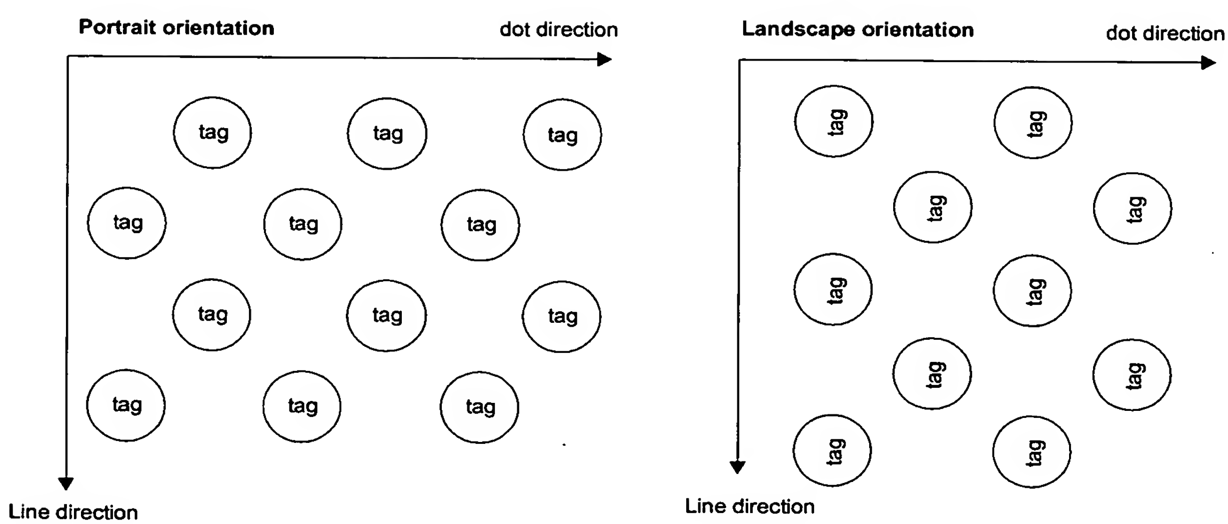


FIG. 182

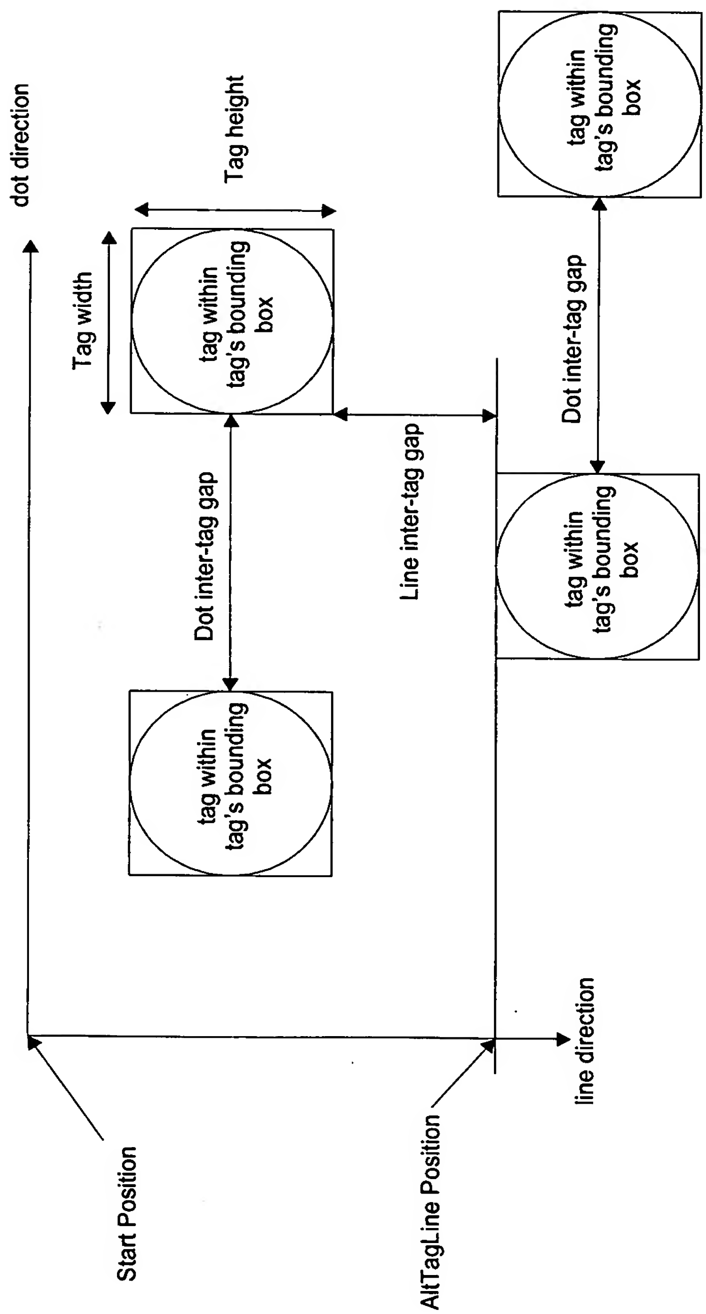
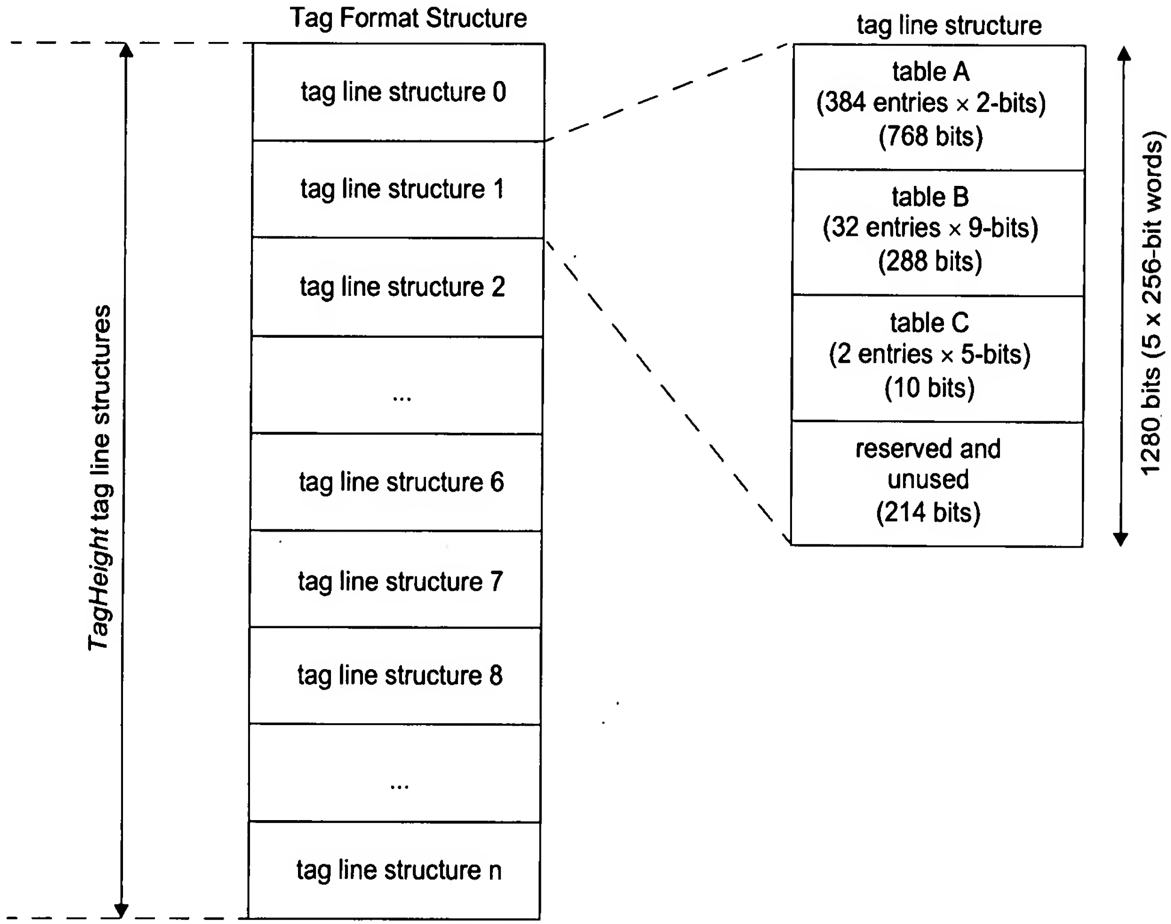

dot direction
Tag height
Tag width
tag within
tag's bounding
box
Dot inter-tag gap
Line inter-tag gap
Dot inter-tag gap
tag within
tag's bounding
box
tag within
tag's bounding
box
tag within
tag's bounding
box
line direction
Start Position
AltTagLine Position

FIG. 183



Tag Format Structure
tag line structure 0
tag line structure 1
tag line structure 2
...
tag line structure 6
tag line structure 7
tag line structure 8
...
tag line structure n
TagHeight tag line structures
tag line structure
table A
(384 entries × 2-bits)
(768 bits)
table B
(32 entries × 9-bits)
(288 bits)
table C
(2 entries × 5-bits)
(10 bits)
reserved and
unused
(214 bits)
1280 bits (5 x 256-bit words)

FIG. 184

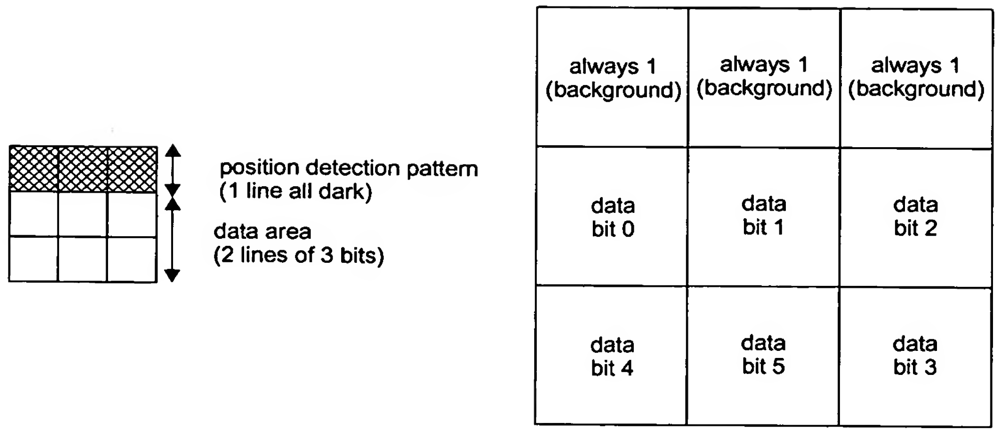


FIG. 185

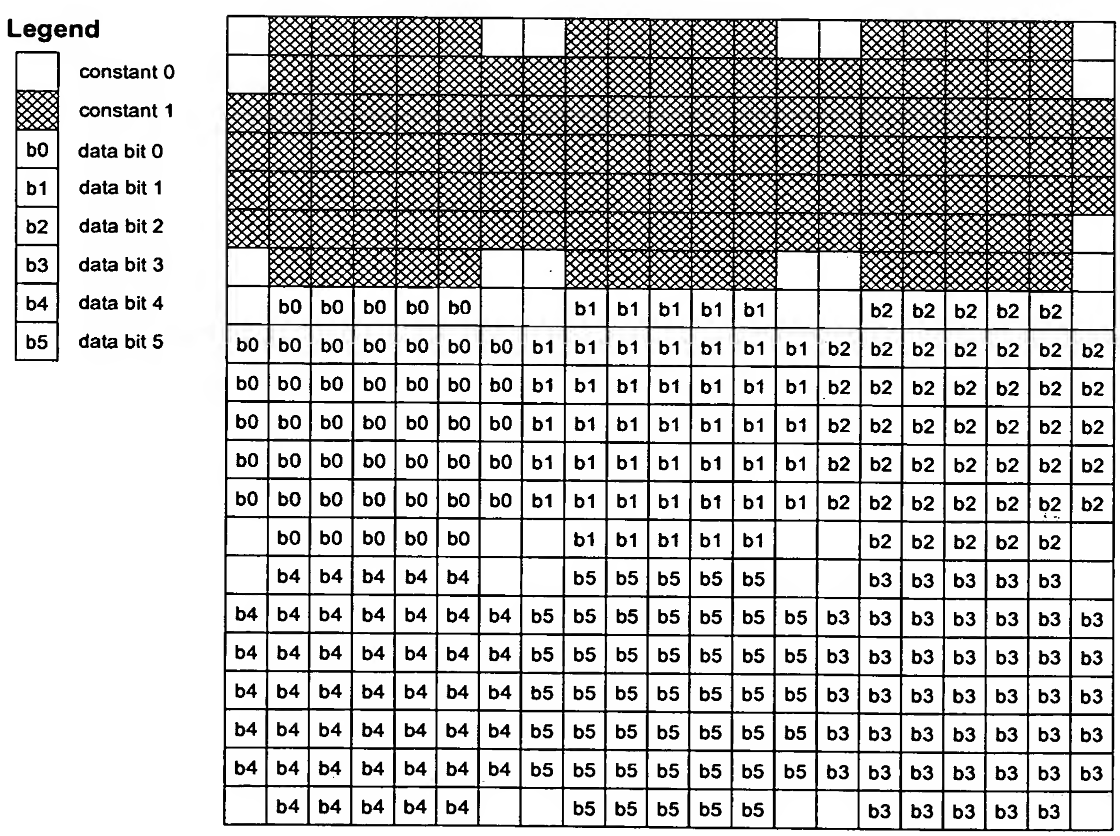


FIG. 186

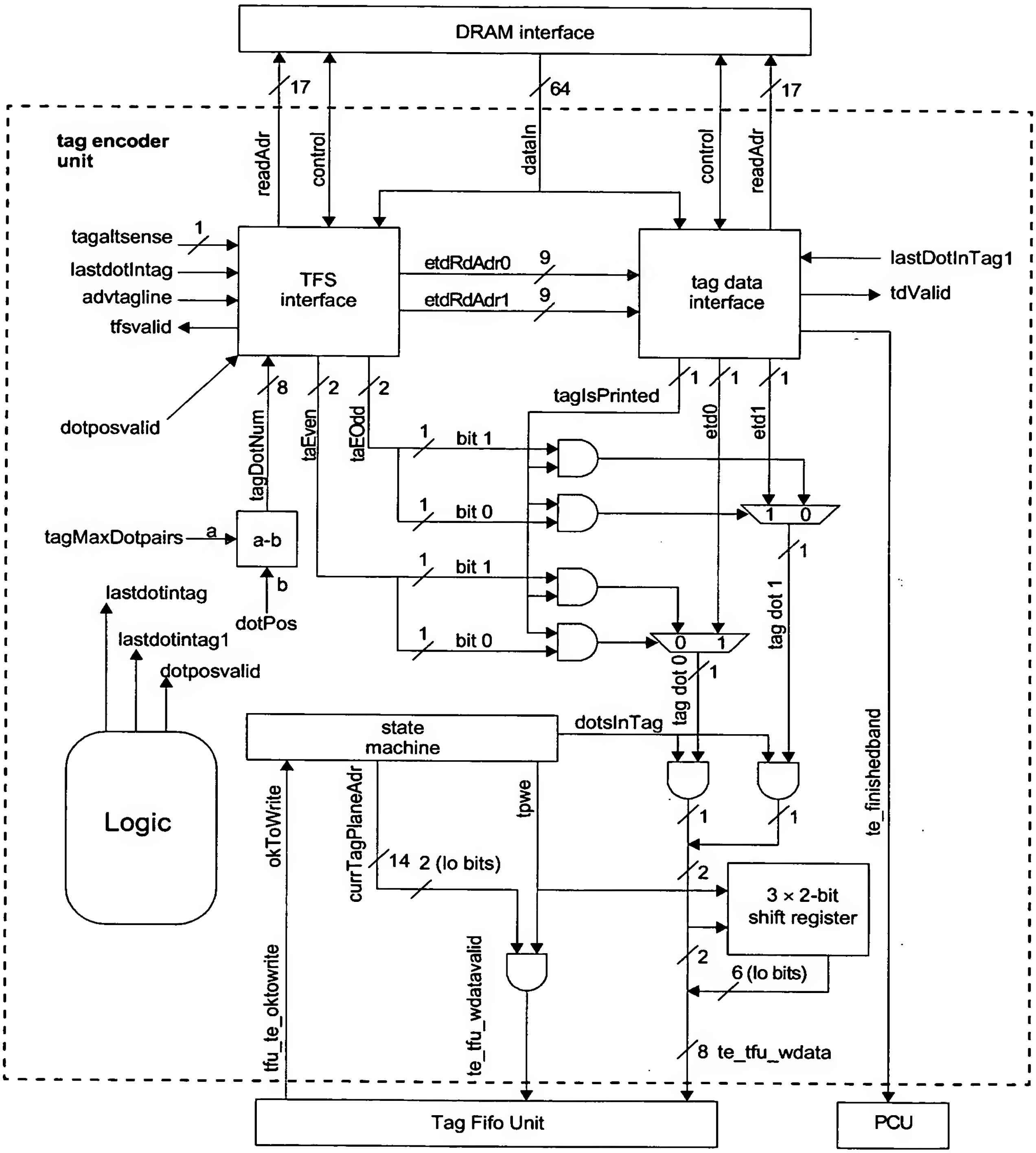

DRAM interface
17
64
17
tag encoder unit
readAdr
control
dataIn
control
readAdr
tagaltsense
1
lastdotIntag
advtagline
tfsvalid
TFS interface
etdRdAdr0
9
etdRdAdr1
9
tag data interface
lastDotInTag1
tdValid
dotposvalid
8
2
2
tagDotNum
taEven
taEOdd
tagIsPrinted
1
1
1
etd0
etd1
bit 1
bit 0
bit 1
bit 0
1
0
0
1
tag dot 1
tag dot 0
tagMaxDotpairs
a
a-b
b
dotPos
lastdotintag
lastdotintag1
dotposvalid
Logic
state machine
dotsInTag
okToWrite
currTagPlaneAdr
tpwe
14
2 (lo bits)
3 × 2-bit shift register
6 (lo bits)
tfu_te_oktowrite
te_tfu_wdatavalid
8
te_tfu_wdata
te_finishedband
Tag Fifo Unit
PCU

FIG. 187

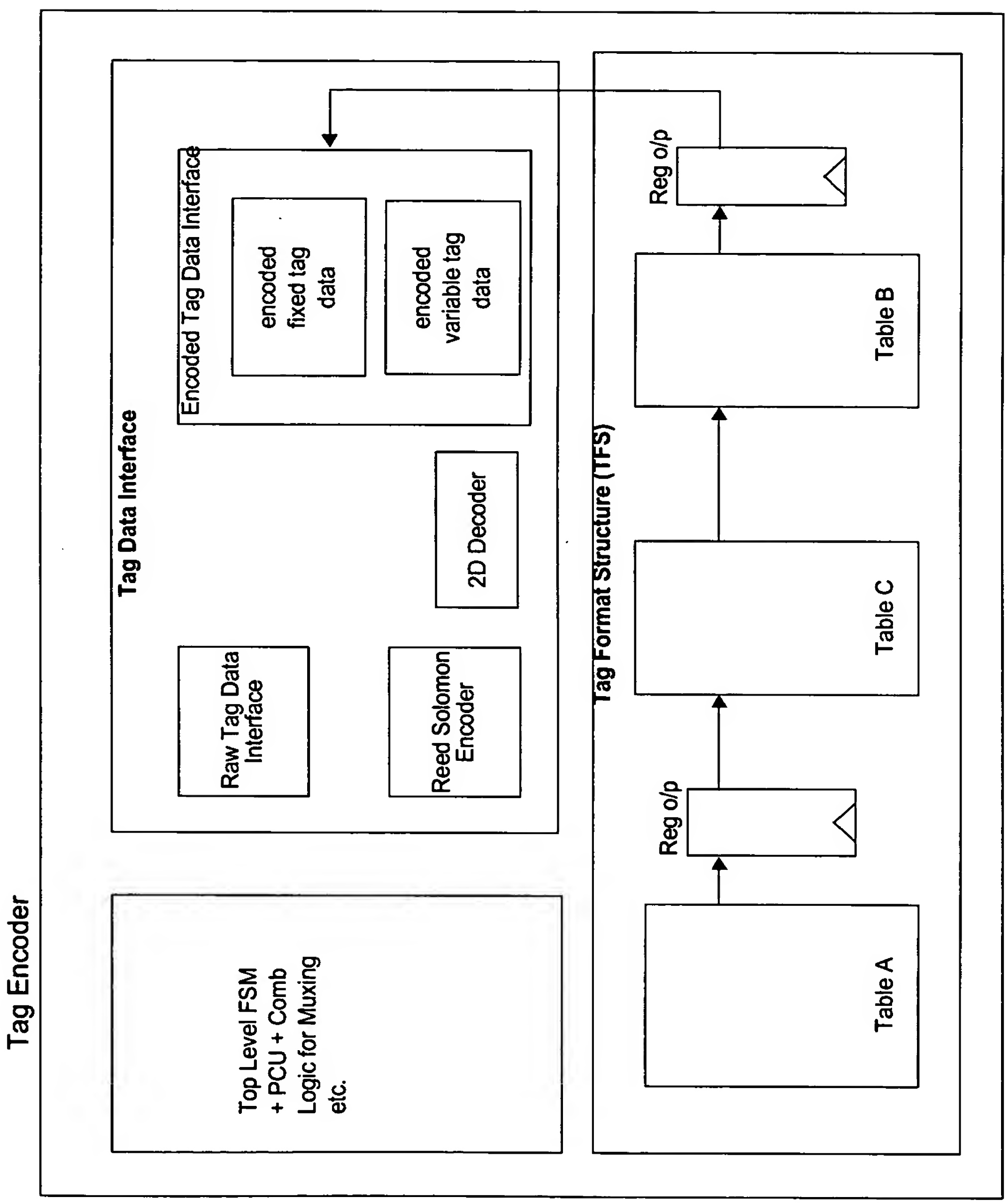

Tag Encoder
Top Level FSM
+ PCU + Comb
Logic for Muxing
etc.
Tag Data Interface
Raw Tag Data
Interface
Reed Solomon
Encoder
2D Decoder
Encoded Tag Data Interface
encoded
fixed tag
data
encoded
variable tag
data
Tag Format Structure (TFS)
Table A
Reg o/p
Table C
Table B
Reg o/p

FIG. 188

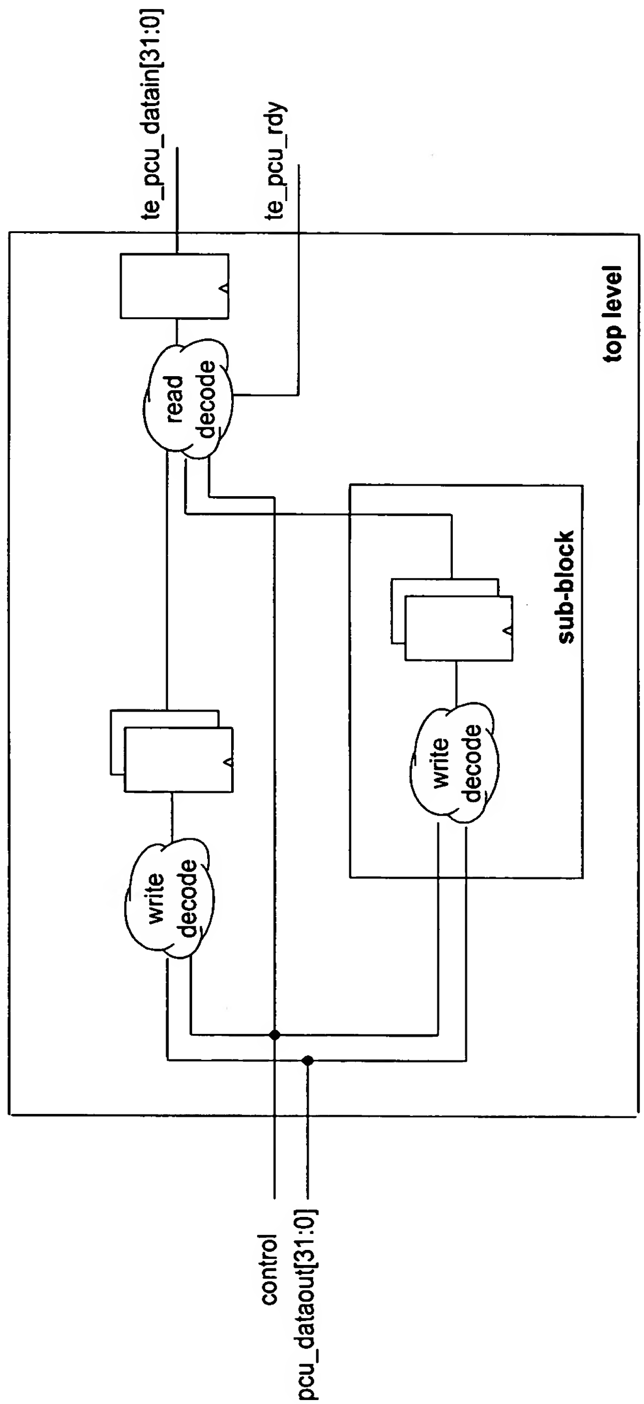

te_pcu_datain[31:0]
te_pcu_rdy
read decode
top level
sub-block
write decode
write decode
control
pcu_dataout[31:0]

FIG. 189

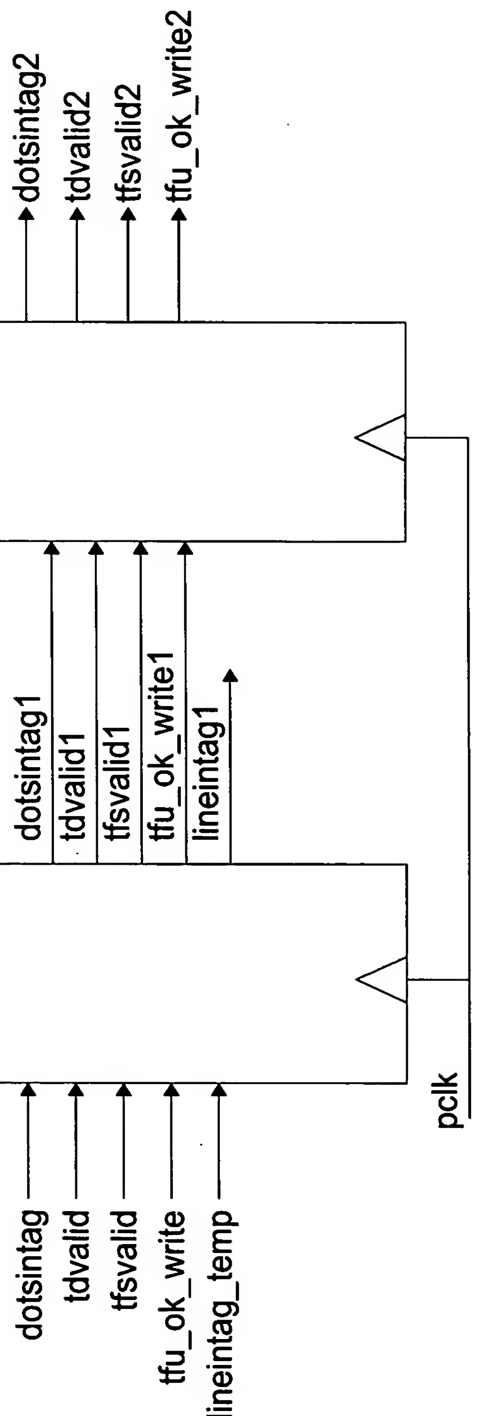

dotsintag2
tdvalid2
tfsvalid2
tfu_ok_write2
dotsintag1
tdvalid1
tfsvalid1
tfu_ok_write1
lineintag1
dotsintag
tdvalid
tfsvalid
tfu_ok_write
lineintag_temp
pclk

FIG. 191

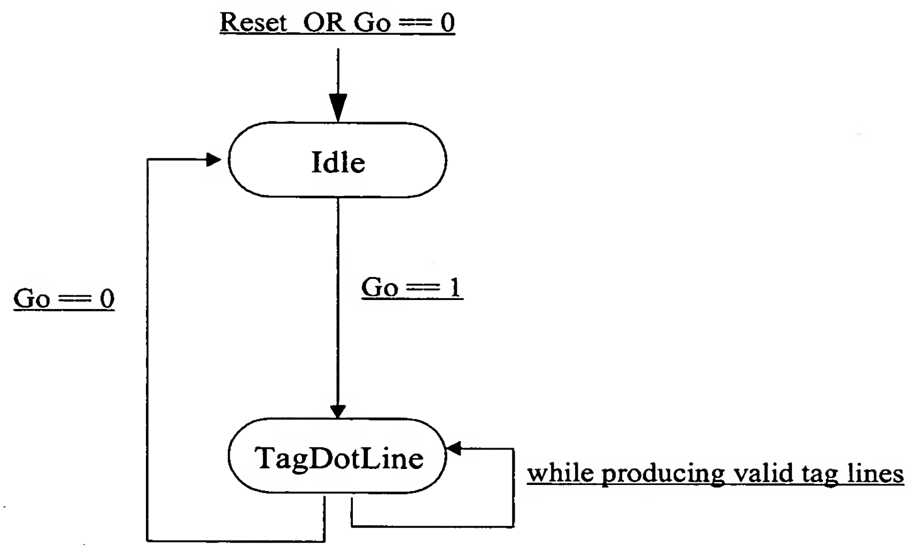


FIG. 190

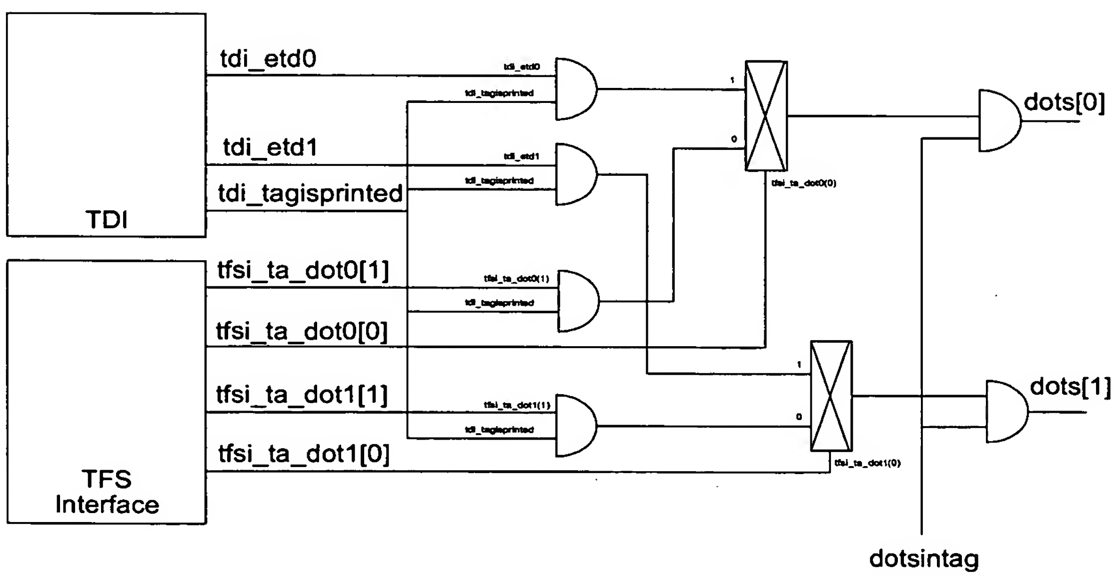


FIG. 192

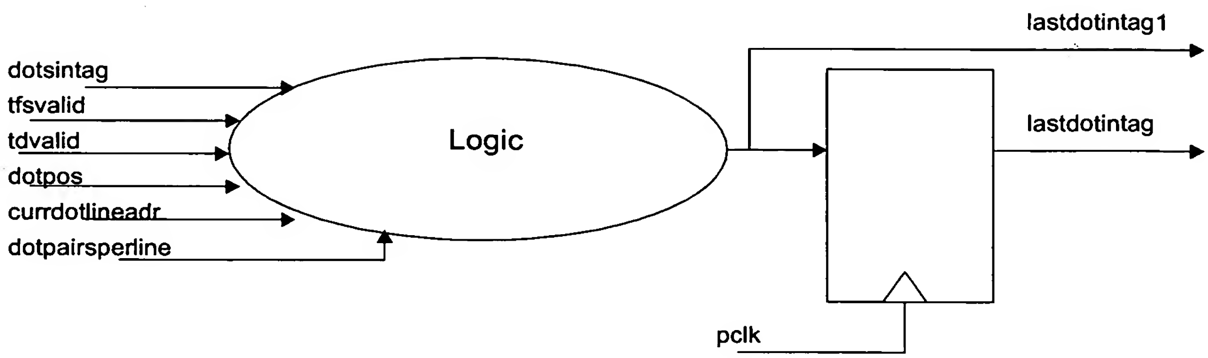


FIG. 193

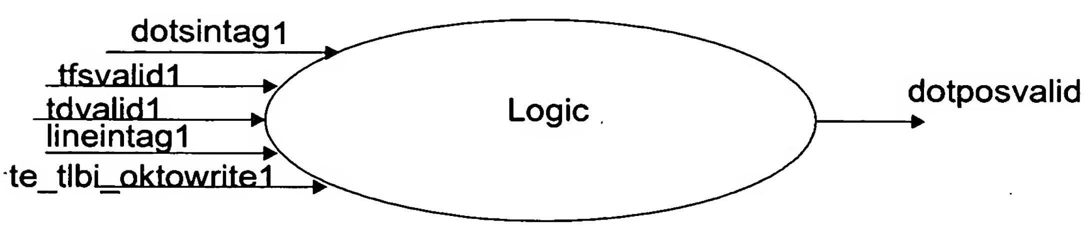


FIG. 194

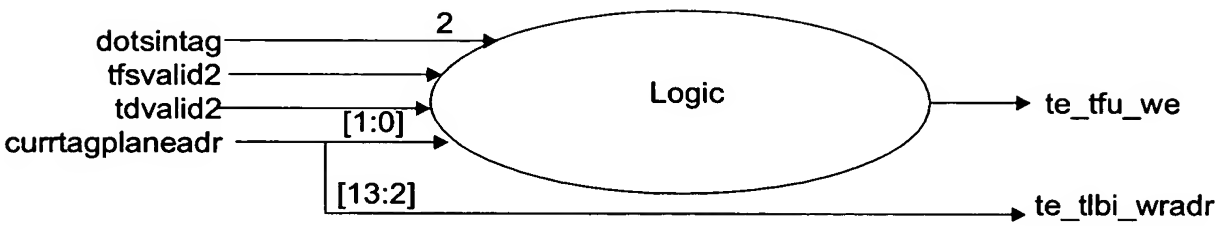


FIG. 195

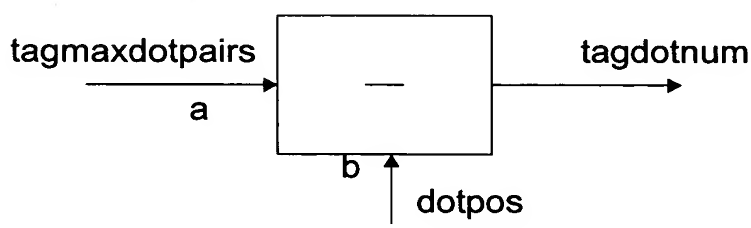


FIG. 196

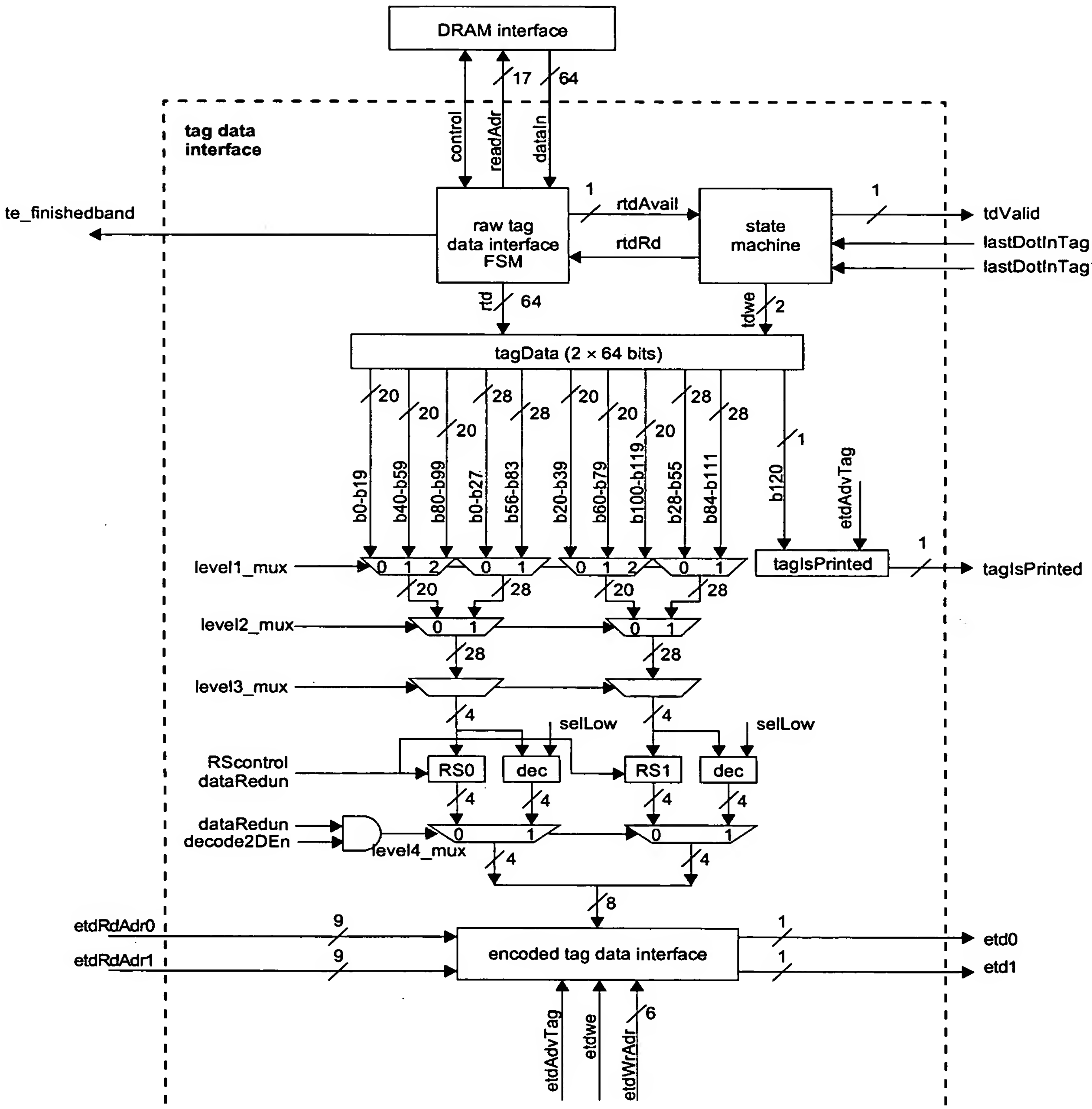
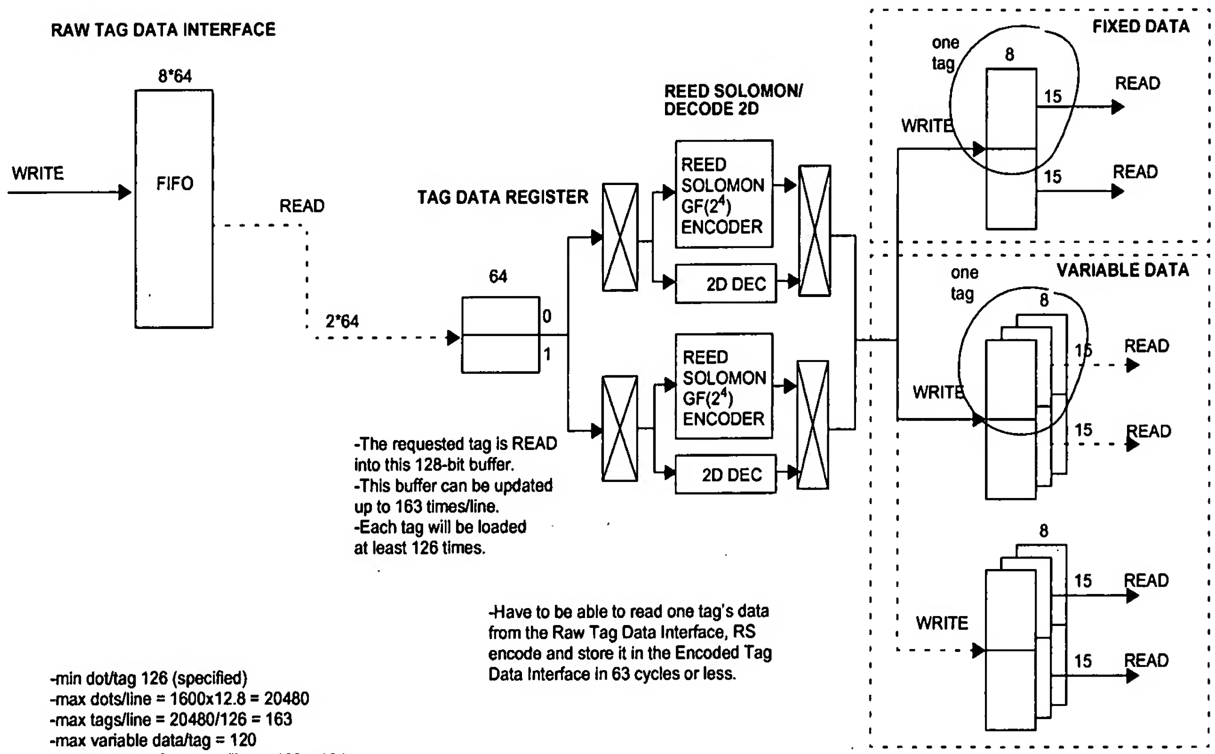

DRAM interface
17
64
tag data
interface
control
readAdr
dataIn
te_finishedband
raw tag
data interface
FSM
1
rtdAvail
rtdRd
state
machine
1
tdValid
lastDotInTag
lastDotInTag1
rtd
64
tdwe
2
tagData (2 × 64 bits)
20
20
20
28
28
20
20
20
28
28
1
b0-b19
b40-b59
b80-b99
b0-b27
b56-b83
b20-b39
b60-b79
b100-b119
b28-b55
b84-b111
b120
etdAdvTag
level1_mux
tagIsPrinted
1
tagIsPrinted
level2_mux
level3_mux
selLow
selLow
RScontrol
dataRedun
RS0
dec
RS1
dec
dataRedun
decode2DEn
level4_mux
8
etdRdAdr0
etdRdAdr1
9
9
encoded tag data interface
1
1
etd0
etd1
etdAdvTag
etdwe
etdWrAdr
6

FIG. 197

ENCODED TAG DATA INTERFACE

-Encoded **fixed data** can be up to 120 bits long
-Use 2 buffers to allow for 2 simultaneously READs in one cycle.
-These stores hold the fixed tag data for 1 tag.
-Total memory = 120x2 = 240 bits



-The requested tag is READ into this 128-bit buffer.
-This buffer can be updated up to 163 times/line.
-Each tag will be loaded at least 126 times.

-Have to be able to read one tag's data from the Raw Tag Data Interface, RS encode and store it in the Encoded Tag Data Interface in 63 cycles or less.

-min dot/tag 126 (specified)
-max dots/line = 1600x12.8 = 20480
-max tags/line = 20480/126 = 163
-max variable data/tag = 120
-max amount of tag data/line = 120 x 164
-Split the 120 tag data bits into 2x64-bits (8 spare bits)
-Max memory needed for 1 line of tag data = 2x64x164 = 656x32
-Divide this in half to allow for simultaneous READ/WRITE
-Once all this data is loaded it will be valid for at least 126 lines.
-From the specification, we must be able to process 2 dots/cycle.
-126 lines contains 20480x126 = 2580480 dots.
-Therefore the data will be updated at most every 1290240 cycles.
-Total memory = 164x2x64 = 20992-bits
-The store uses 9-bit addressing. Bit-9 indicates which buffer.
-Once printing has started each half buffer has 1/2 a line in which to be loaded
i.e. for a 12,8 inch line it has 10240 dots or 5120 cycles
for an 8 inch line it has 6400 dots or 3200 cycles

-Encoded **variable data** can be up to 360 bits long
-Use 2 buffers to allow for 2 simultaneously READs in one cycle.
-Use 2 buffers to allow for simultaneously READ/WRITE
-Total memory = 360x2x2 = 1280 bits
-Min tag width = 126 dots
so the fastest that 1 tag can be read = 126/2 = 63 cycles

FIG. 198

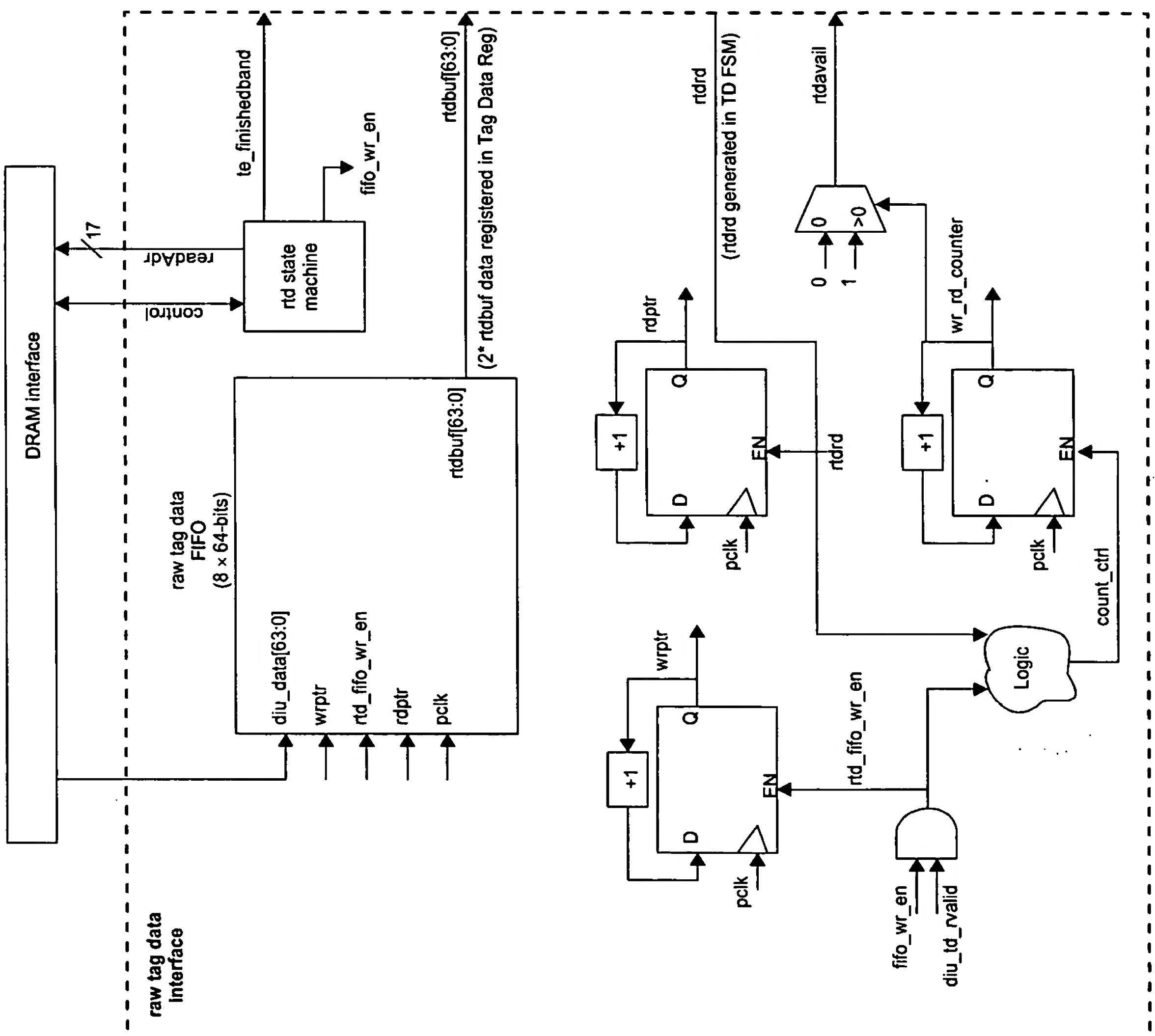

DRAM interface
raw tag data interface
readAdr
17
control
rtd state machine
te_finishedband
fifo_wr_en
rtdbuf[63:0]
(2* rtdbuf data registered in Tag Data Reg)
raw tag data FIFO (8 × 64-bits)
diu_data[63:0]
wrptr
rtd_fifo_wr_en
rdptr
pclk
rtdbuf[63:0]
rtdrd
(rtdrd generated in TD FSM)
rtdavail
0
1
>0
wr_rd_counter
+1
D
Q
EN
pclk
rdptr
rtdrd
wrptr
rtd_fifo_wr_en
Logic
count_ctrl
fifo_wr_en
diu_td_rvalid

FIG. 199

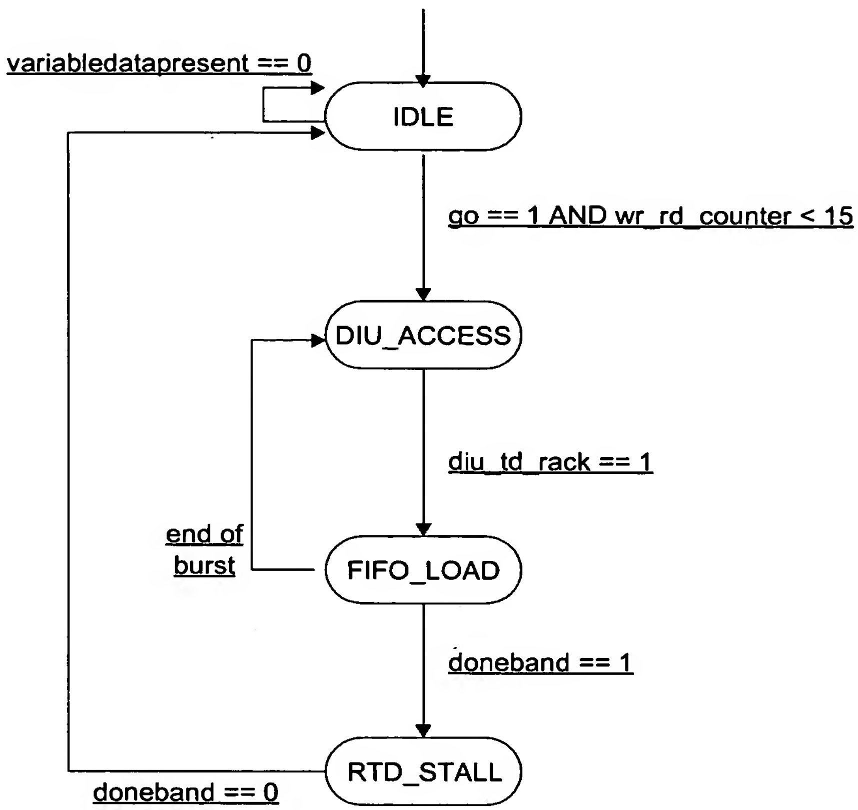


FIG. 200

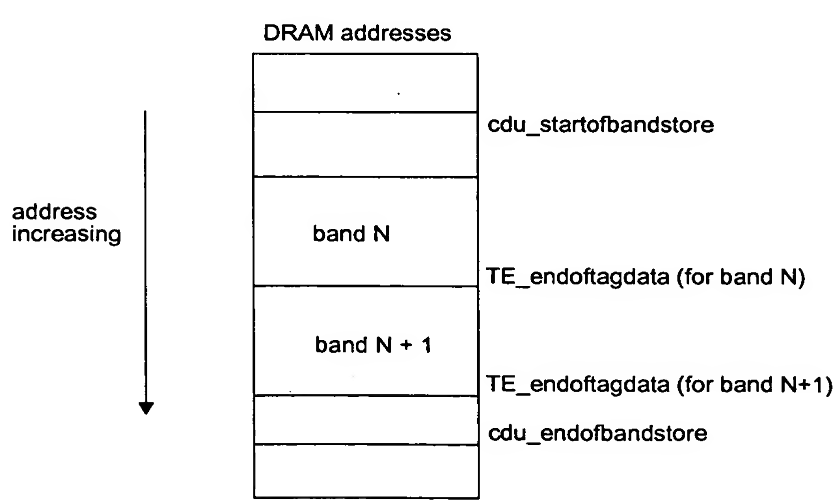


FIG. 201

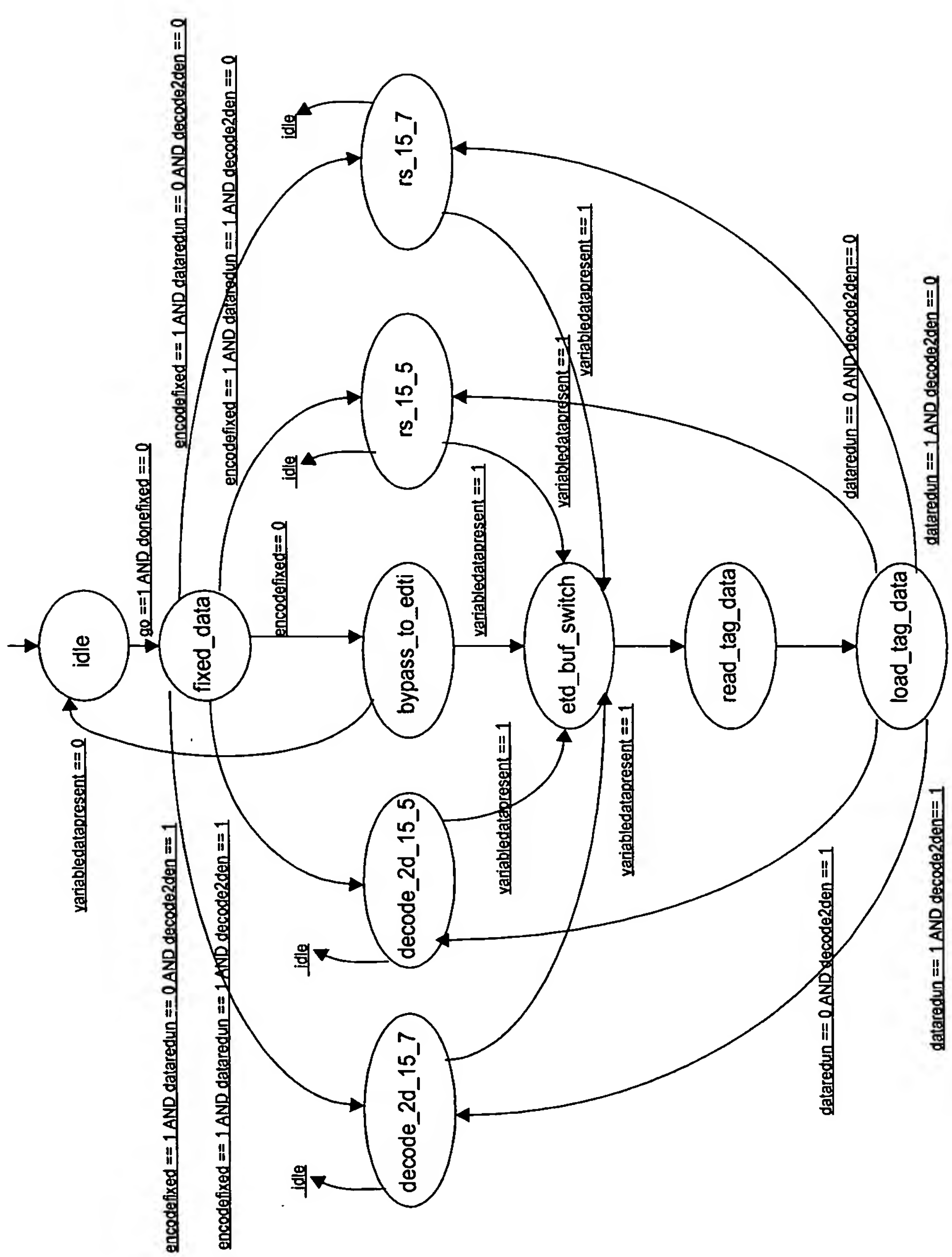

idle
go ==1 AND donefixed == 0
fixed_data
encodefixed == 1 AND dataredun == 0 AND decode2den == 0
encodefixed == 1 AND dataredun == 1 AND decode2den == 0
encodefixed== 0
bypass_to_edti
rs_15_5
rs_15_7
idle
idle
variabledatapresent == 1
variabledatapresent == 1
variabledatapresent == 1
variabledatapresent == 0
etd_buf_switch
read_tag_data
load_tag_data
decode_2d_15_5
decode_2d_15_7
idle
idle
variabledatapresent == 1
variabledatapresent == 1
encodefixed == 1 AND dataredun == 0 AND decode2den == 1
encodefixed == 1 AND dataredun == 1 AND decode2den == 1
dataredun == 0 AND decode2den== 0
dataredun == 1 AND decode2den == 0
dataredun == 0 AND decode2den == 1
dataredun == 1 AND decode2den== 1

FIG. 202

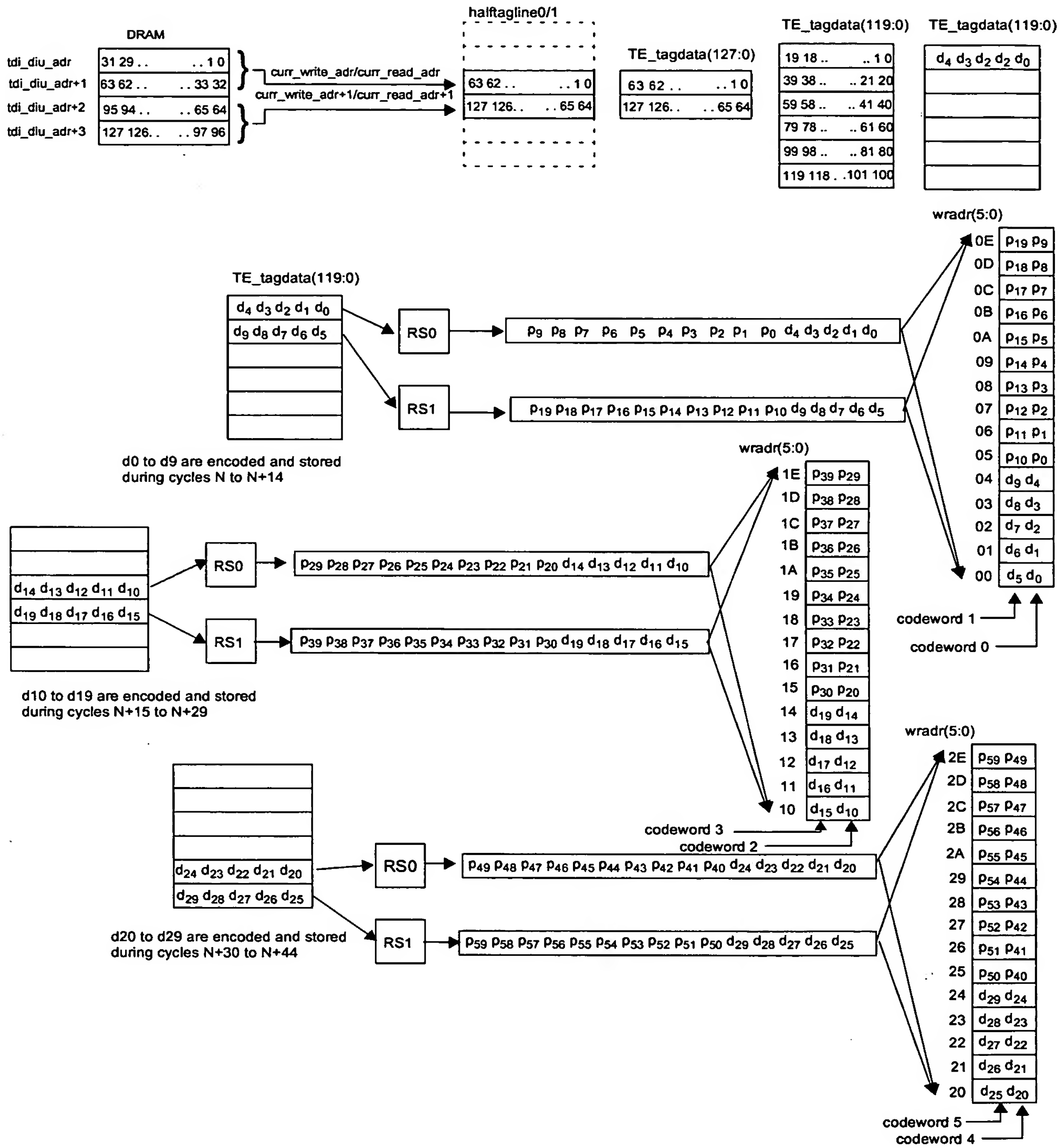
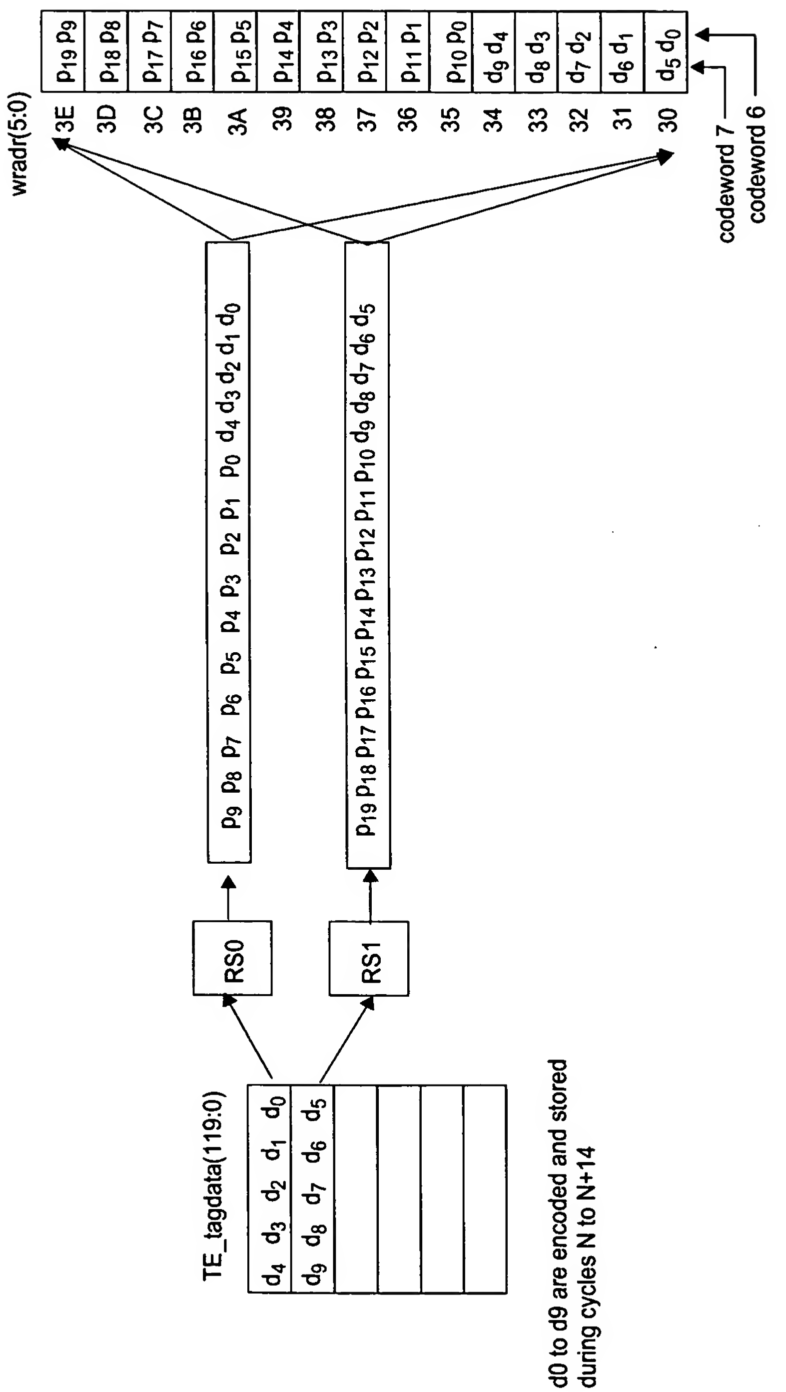

DRAM
tdi_diu_adr
tdi_diu_adr+1
tdi_diu_adr+2
tdi_diu_adr+3
31 29 .. ..1 0
63 62 .. .. 33 32
95 94 .. .. 65 64
127 126.. ..97 96
curr_write_adr/curr_read_adr
curr_write_adr+1/curr_read_adr+1
halftagline0/1
63 62 .. ..1 0
127 126.. ..65 64
TE_tagdata(127:0)
63 62 .. ..1 0
127 126.. ..65 64
TE_tagdata(119:0)
19 18 .. .. 1 0
39 38 .. .. 21 20
59 58 .. .. 41 40
79 78 .. .. 61 60
99 98 .. .. 81 80
119 118 . .101 100
TE_tagdata(119:0)
d4 d3 d2 d2 d0
TE_tagdata(119:0)
d4 d3 d2 d1 d0
d9 d8 d7 d6 d5
RS0
RS1
p9 p8 p7 p6 p5 p4 p3 p2 p1 p0 d4 d3 d2 d1 d0
p19 p18 p17 p16 p15 p14 p13 p12 p11 p10 d9 d8 d7 d6 d5
d0 to d9 are encoded and stored
during cycles N to N+14
wradr(5:0)
0E p19 p9
0D p18 p8
0C p17 p7
0B p16 p6
0A p15 p5
09 p14 p4
08 p13 p3
07 p12 p2
06 p11 p1
05 p10 p0
04 d9 d4
03 d8 d3
02 d7 d2
01 d6 d1
00 d5 d0
codeword 1
codeword 0
d14 d13 d12 d11 d10
d19 d18 d17 d16 d15
p29 p28 p27 p26 p25 p24 p23 p22 p21 p20 d14 d13 d12 d11 d10
p39 p38 p37 p36 p35 p34 p33 p32 p31 p30 d19 d18 d17 d16 d15
d10 to d19 are encoded and stored
during cycles N+15 to N+29
wradr(5:0)
1E p39 p29
1D p38 p28
1C p37 p27
1B p36 p26
1A p35 p25
19 p34 p24
18 p33 p23
17 p32 p22
16 p31 p21
15 p30 p20
14 d19 d14
13 d18 d13
12 d17 d12
11 d16 d11
10 d15 d10
codeword 3
codeword 2
d24 d23 d22 d21 d20
d29 d28 d27 d26 d25
p49 p48 p47 p46 p45 p44 p43 p42 p41 p40 d24 d23 d22 d21 d20
p59 p58 p57 p56 p55 p54 p53 p52 p51 p50 d29 d28 d27 d26 d25
d20 to d29 are encoded and stored
during cycles N+30 to N+44
wradr(5:0)
2E p59 p49
2D p58 p48
2C p57 p47
2B p56 p46
2A p55 p45
29 p54 p44
28 p53 p43
27 p52 p42
26 p51 p41
25 p50 p40
24 d29 d24
23 d28 d23
22 d27 d22
21 d26 d21
20 d25 d20
codeword 5
codeword 4

FIG. 203



TE_tagdata(119:0)
d4 d3 d2 d1 d0
d9 d8 d7 d6 d5
d0 to d9 are encoded and stored during cycles N to N+14
RS0
RS1
p9 p8 p7 p6 p5 p4 p3 p2 p1 p0 d4 d3 d2 d1 d0
p19 p18 p17 p16 p15 p14 p13 p12 p11 p10 d9 d8 d7 d6 d5
wradr(5:0)
3E p19 p9
3D p18 p8
3C p17 p7
3B p16 p6
3A p15 p5
39 p14 p4
38 p13 p3
37 p12 p2
36 p11 p1
35 p10 p0
34 d9 d4
33 d8 d3
32 d7 d2
31 d6 d1
30 d5 d0
codeword 7
codeword 6

FIG. 204

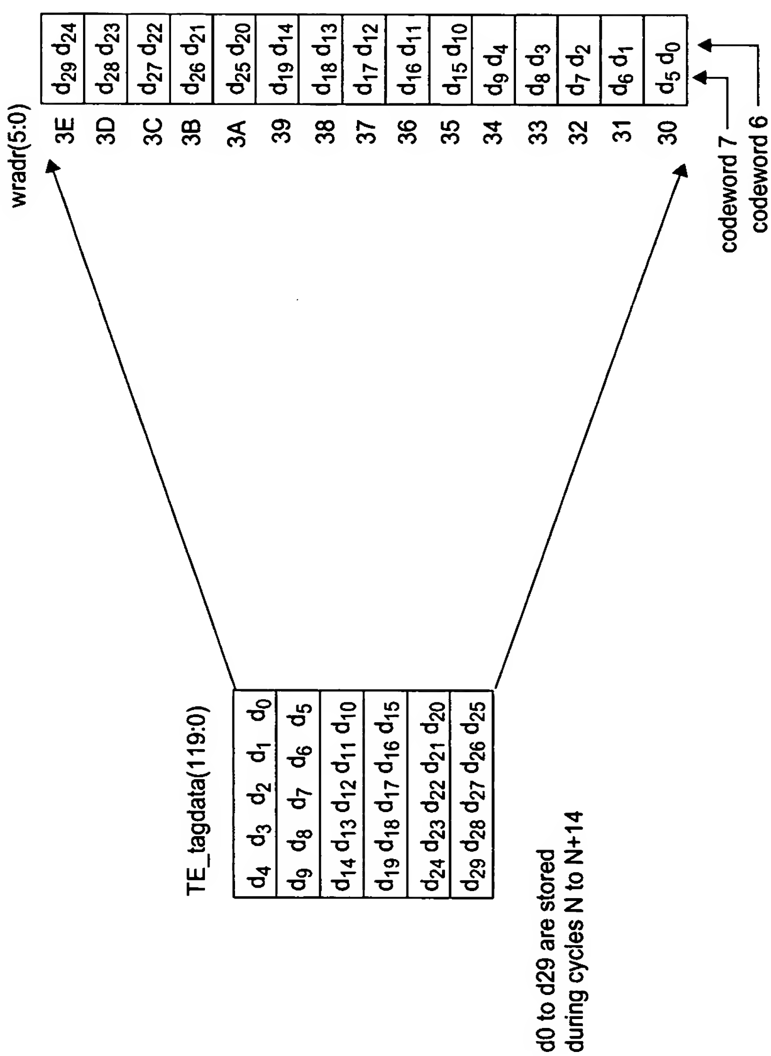
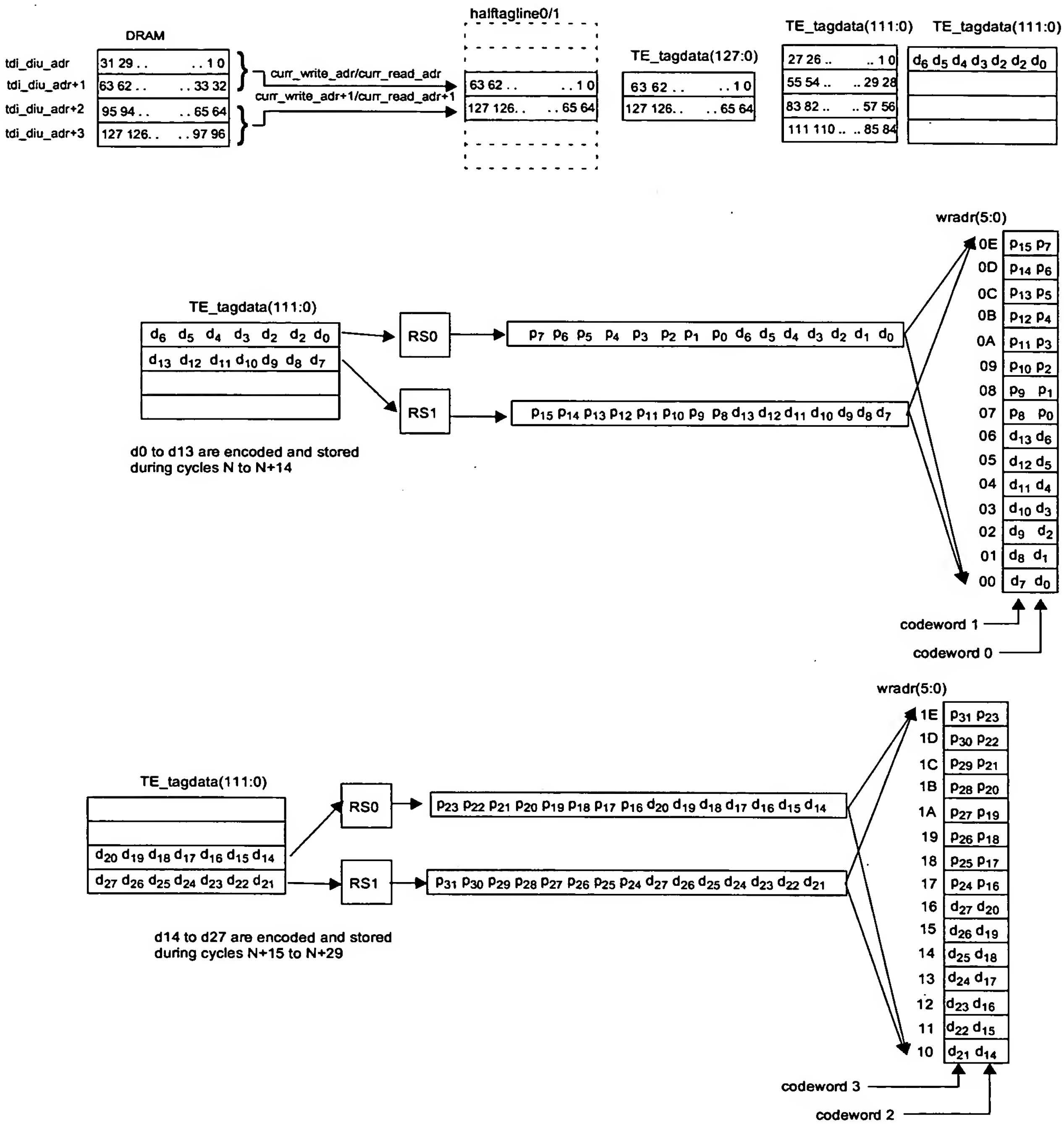
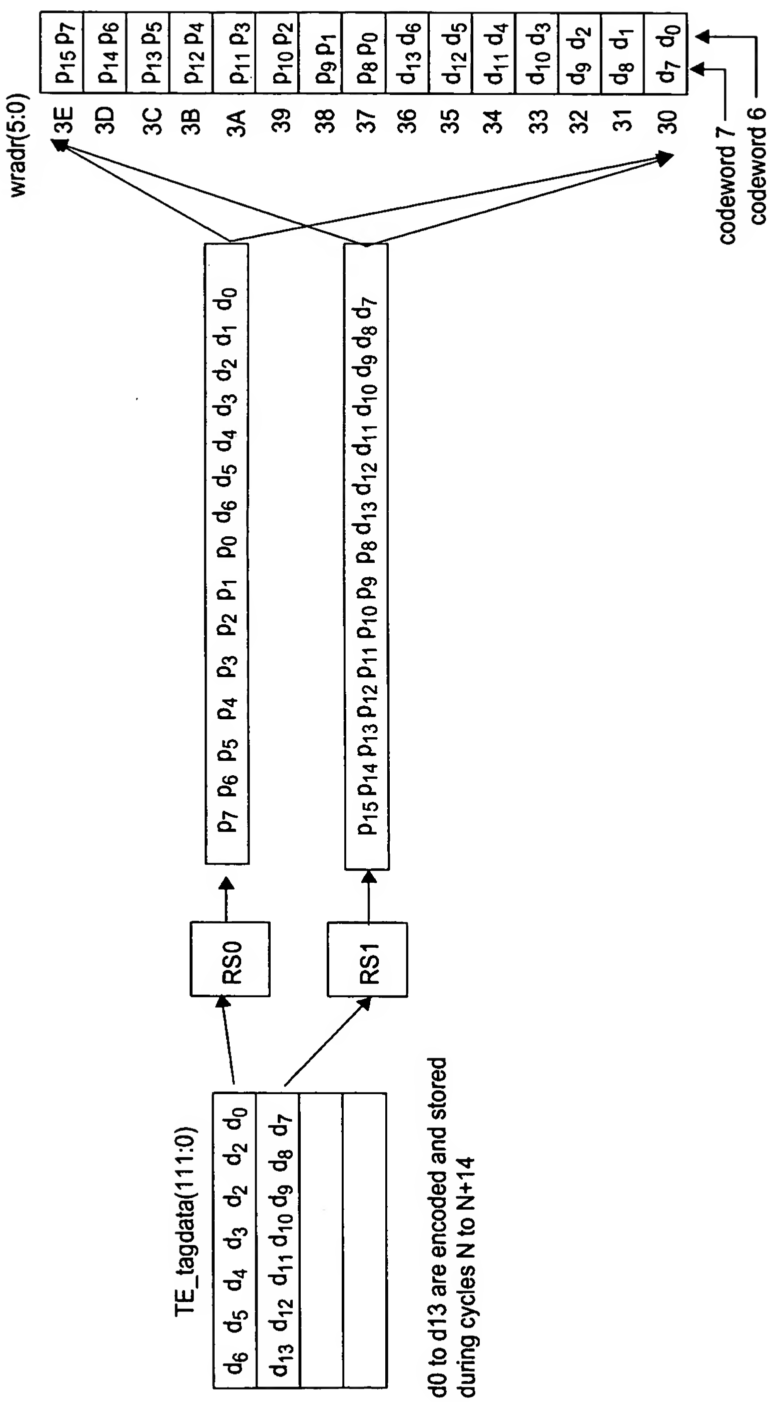

TE_tagdata(119:0)
d4 d3 d2 d1 d0
d9 d8 d7 d6 d5
d14 d13 d12 d11 d10
d19 d18 d17 d16 d15
d24 d23 d22 d21 d20
d29 d28 d27 d26 d25
d0 to d29 are stored
during cycles N to N+14
wradr(5:0)
3E d29 d24
3D d28 d23
3C d27 d22
3B d26 d21
3A d25 d20
39 d19 d14
38 d18 d13
37 d17 d12
36 d16 d11
35 d15 d10
34 d9 d4
33 d8 d3
32 d7 d2
31 d6 d1
30 d5 d0
codeword 7
codeword 6

FIG. 205



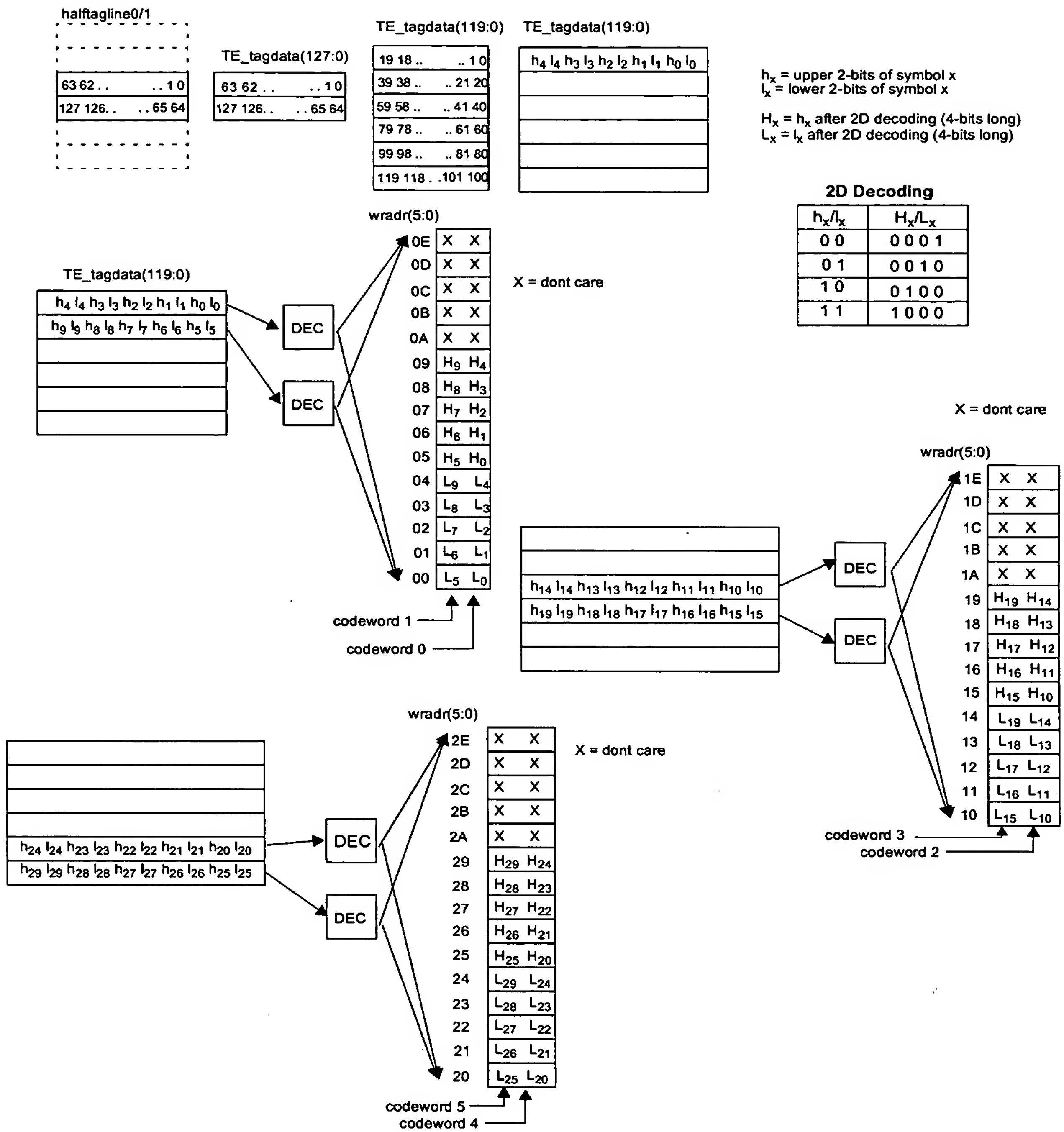
halftagline0/1
DRAM
tdi_diu_adr
tdi_diu_adr+1
tdi_diu_adr+2
tdi_diu_adr+3
31 29 .. ..1 0
63 62 .. ..33 32
95 94 .. ..65 64
127 126.. ..97 96
curr_write_adr/curr_read_adr
curr_write_adr+1/curr_read_adr+1
63 62 .. ..1 0
127 126.. ..65 64
TE_tagdata(127:0)
TE_tagdata(111:0)
27 26 .. ..1 0
55 54 .. ..29 28
83 82 .. ..57 56
111 110 .. ..85 84
d6 d5 d4 d3 d2 d2 d0
d13 d12 d11 d10 d9 d8 d7
RS0
RS1
p7 p6 p5 p4 p3 p2 p1 p0 d6 d5 d4 d3 d2 d1 d0
p15 p14 p13 p12 p11 p10 p9 p8 d13 d12 d11 d10 d9 d8 d7
d0 to d13 are encoded and stored during cycles N to N+14
wradr(5:0)
codeword 1
codeword 0
d20 d19 d18 d17 d16 d15 d14
d27 d26 d25 d24 d23 d22 d21
p23 p22 p21 p20 p19 p18 p17 p16 d20 d19 d18 d17 d16 d15 d14
p31 p30 p29 p28 p27 p26 p25 p24 d27 d26 d25 d24 d23 d22 d21
d14 to d27 are encoded and stored during cycles N+15 to N+29
codeword 3
codeword 2

FIG. 206



wradr(5:0)
3E 3D 3C 3B 3A 39 38 37 36 35 34 33 32 31 30
p15 p7
p14 p6
p13 p5
p12 p4
p11 p3
p10 p2
p9 p1
p8 p0
d13 d6
d12 d5
d11 d4
d10 d3
d9 d2
d8 d1
d7 d0
codeword 7
codeword 6
p7 p6 p5 p4 p3 p2 p1 p0 d6 d5 d4 d3 d2 d1 d0
p15 p14 p13 p12 p11 p10 p9 p8 d13 d12 d11 d10 d9 d8 d7
RS0
RS1
TE_tagdata(111:0)
d6 d5 d4 d3 d2 d2 d0
d13 d12 d11 d10 d9 d8 d7
d0 to d13 are encoded and stored during cycles N to N+14

FIG. 207



h_x/l_x	H_x/L_x
0 0	0 0 0 1
0 1	0 0 1 0
1 0	0 1 0 0
1 1	1 0 0 0

FIG. 208

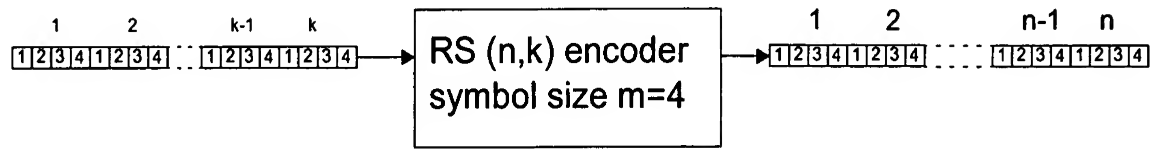


FIG. 209

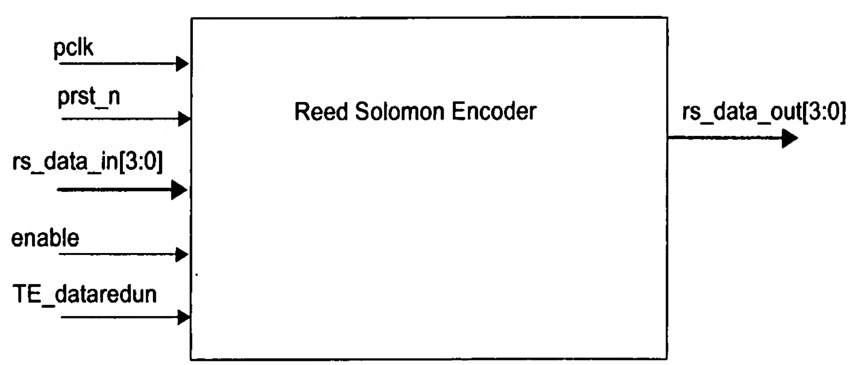


FIG. 210

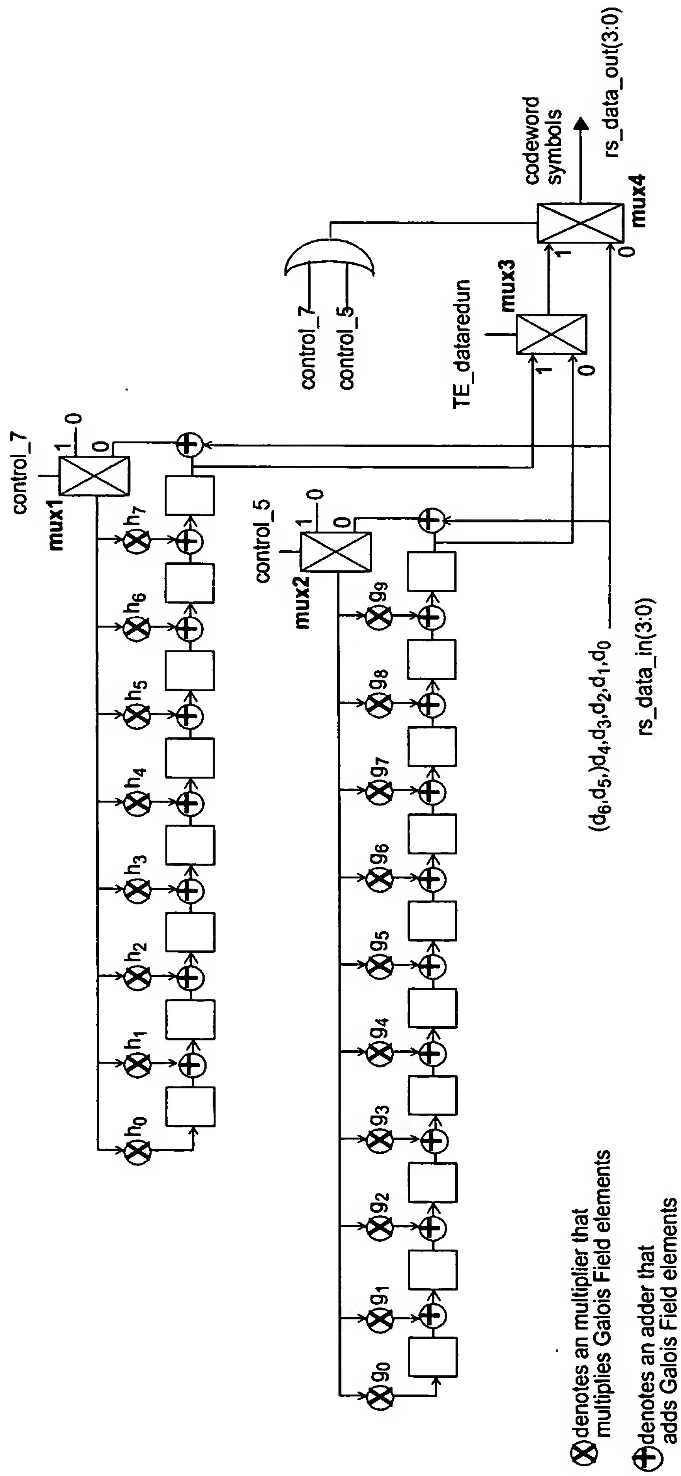

control_7
mux1
1
0
h_0
h_1
h_2
h_3
h_4
h_5
h_6
h_7
control_5
mux2
1
0
g_0
g_1
g_2
g_3
g_4
g_5
g_6
g_7
g_8
g_9
TE_dataredun
mux3
control_7
control_5
codeword symbols
rs_data_out(3:0)
mux4
$(d_6,d_5,)d_4,d_3,d_2,d_1,d_0$
rs_data_in(3:0)
denotes an multiplier that multiplies Galois Field elements
denotes an adder that adds Galois Field elements

FIG. 211

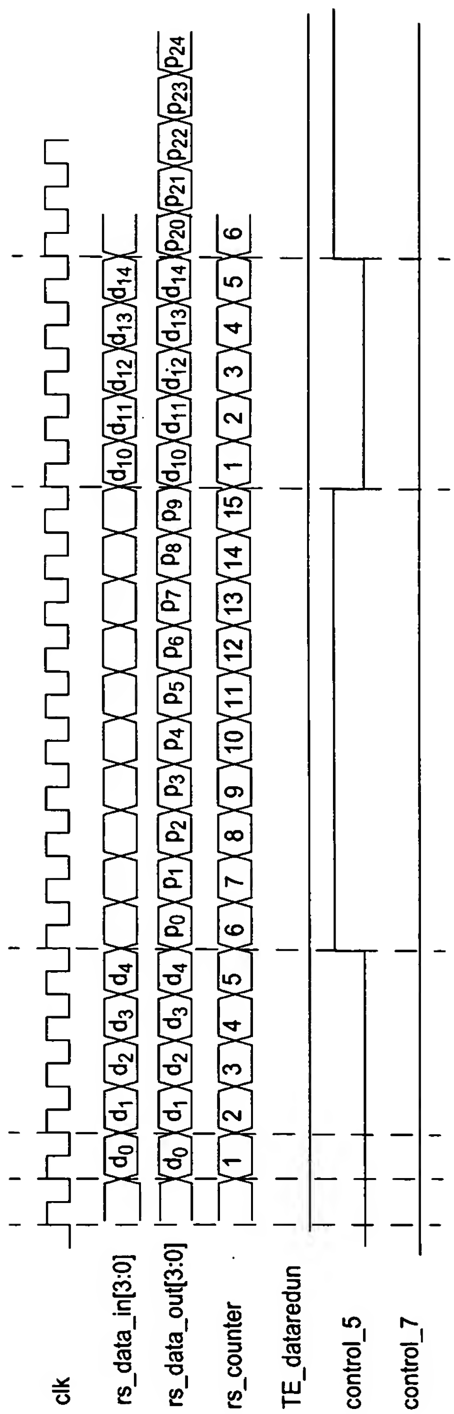

clk
rs_data_in[3:0]
rs_data_out[3:0]
rs_counter
TE_dataredun
control_5
control_7

FIG. 212

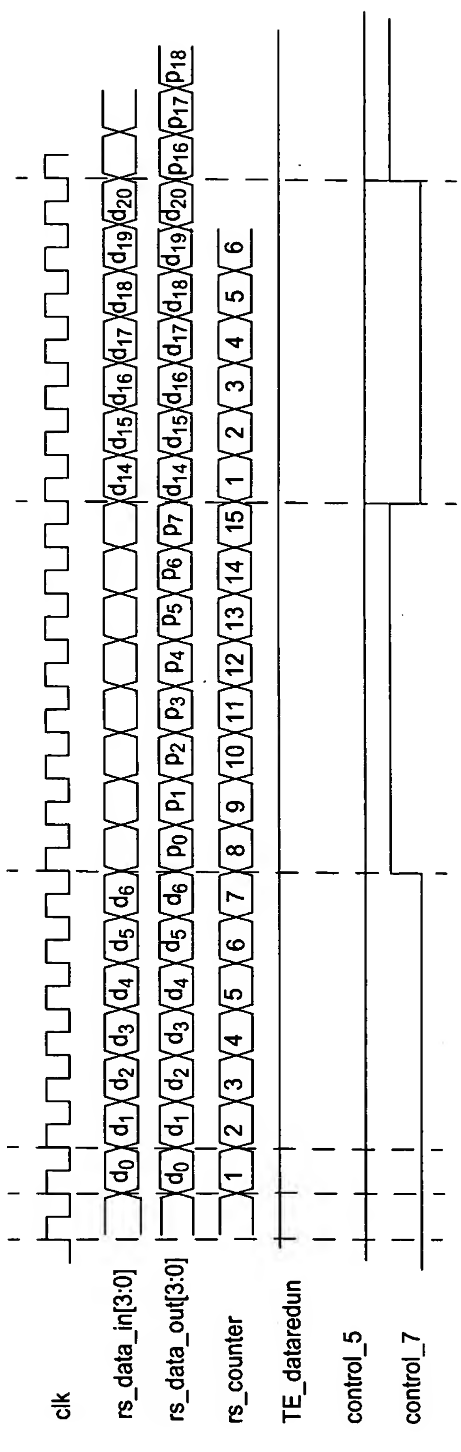

clk
rs_data_in[3:0]
rs_data_out[3:0]
rs_counter
TE_dataredun
control_5
control_7
d_0 d_1 d_2 d_3 d_4 d_5 d_6 d_{14} d_{15} d_{16} d_{17} d_{18} d_{19} d_{20}
d_0 d_1 d_2 d_3 d_4 d_5 d_6 p_0 p_1 p_2 p_3 p_4 p_5 p_6 p_7 d_{14} d_{15} d_{16} d_{17} d_{18} d_{19} d_{20} p_{16} p_{17} p_{18}
1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 1 2 3 4 5 6

FIG. 213

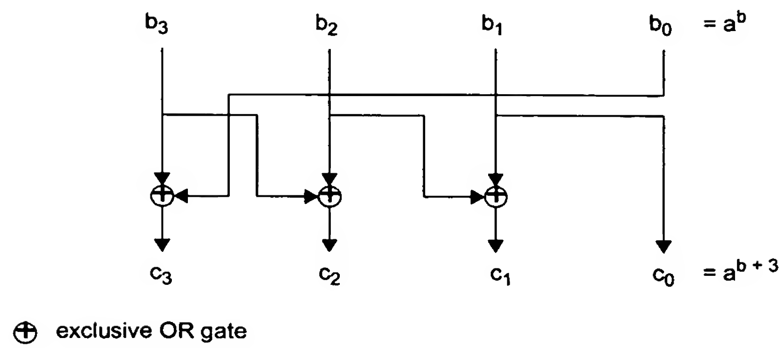


FIG. 214

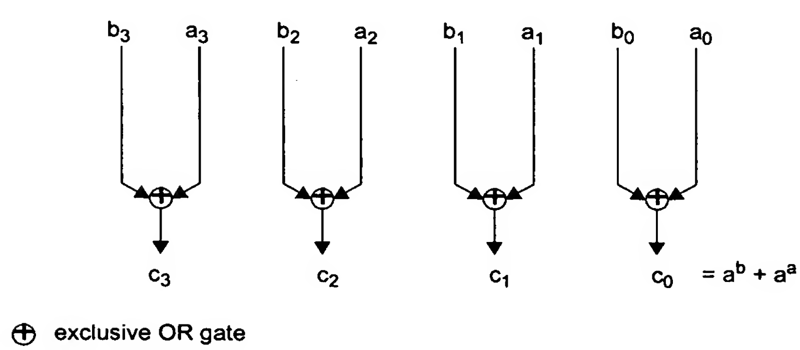


FIG. 215

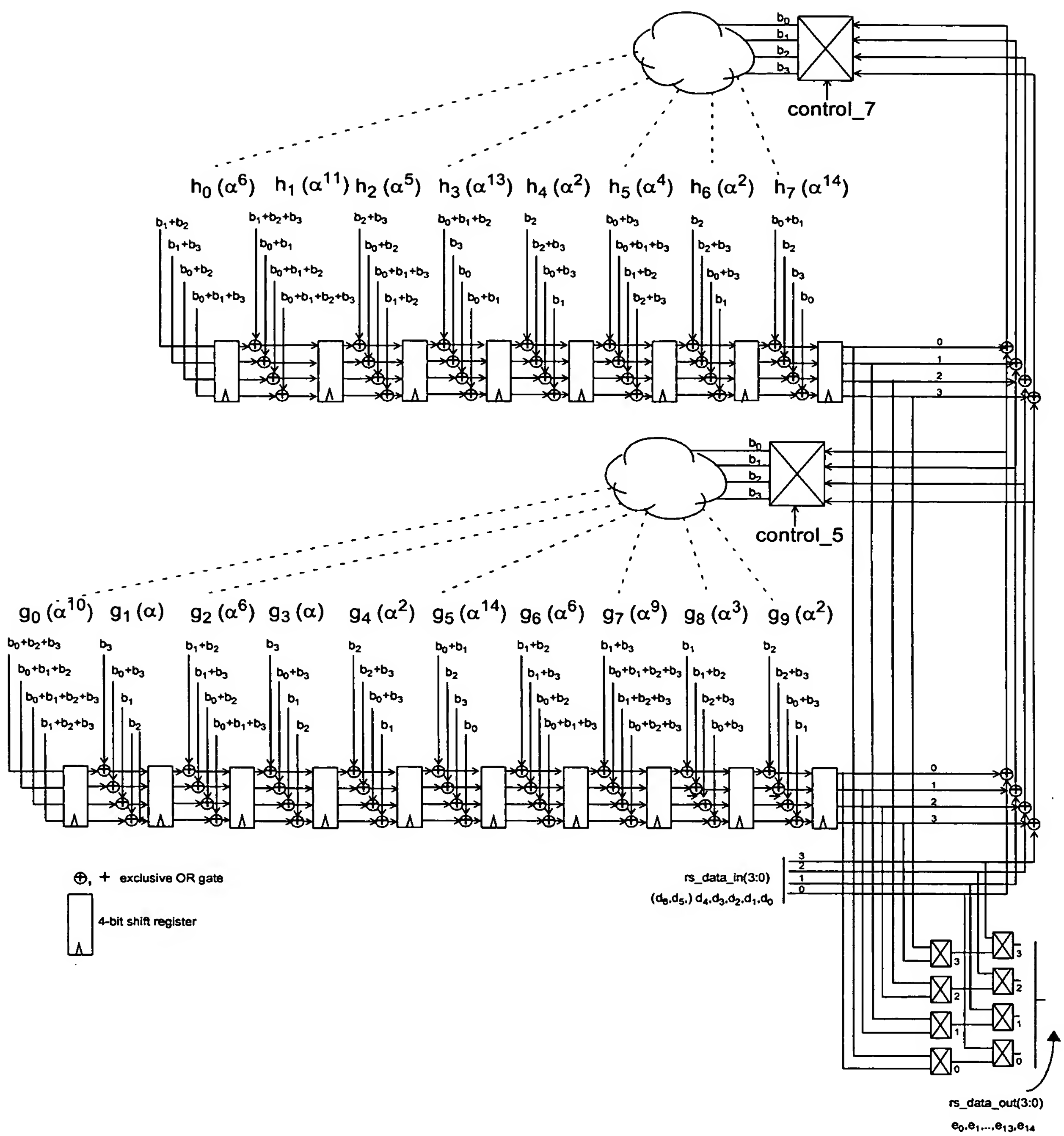

control_7
control_5
exclusive OR gate
4-bit shift register
rs_data_in(3:0)
rs_data_out(3:0)

FIG. 216

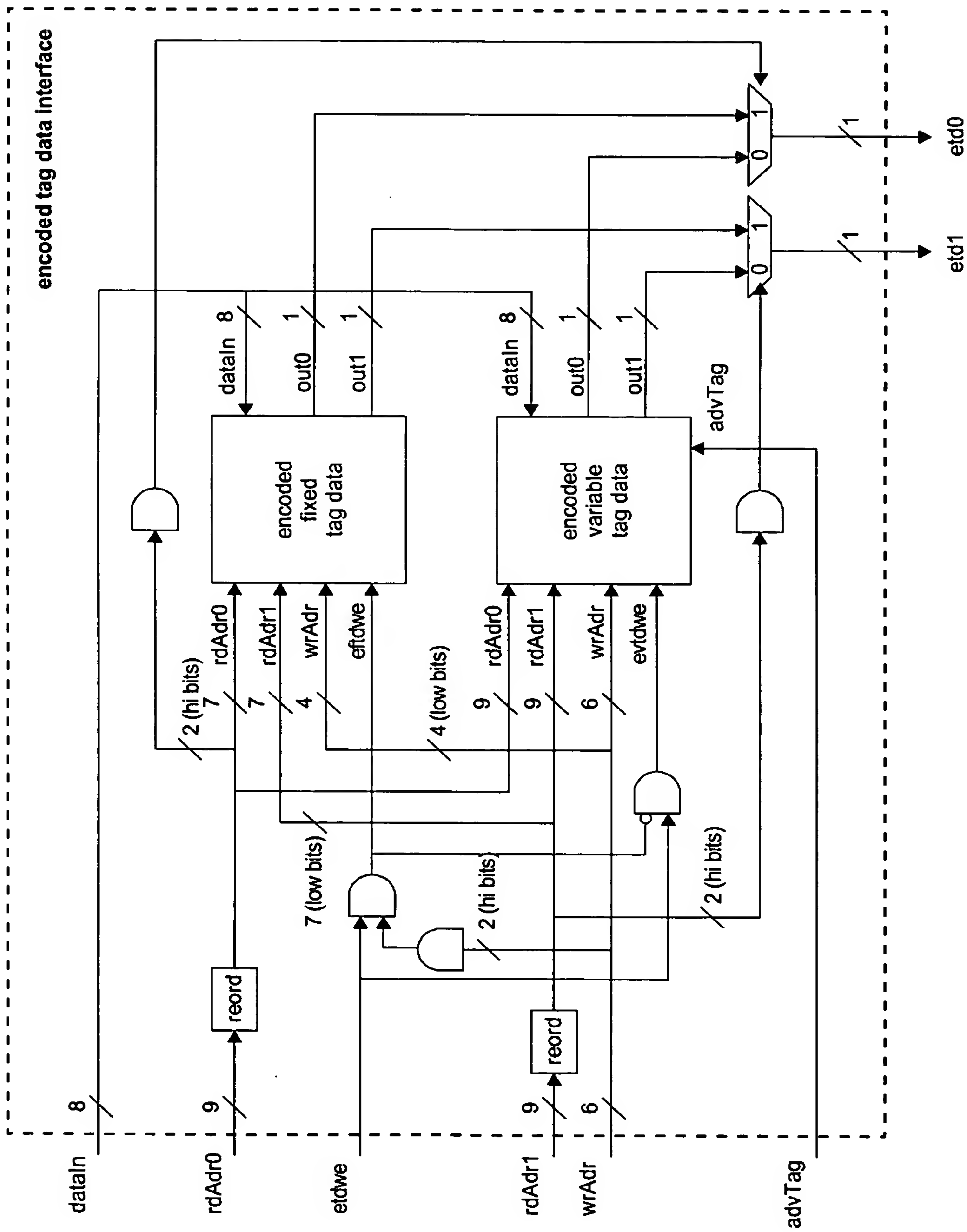

encoded tag data interface
dataIn
rdAdr0
etdwe
rdAdr1
wrAdr
advTag
8
9
9
6
reord
reord
7 (low bits)
2 (hi bits)
2 (hi bits)
2 (hi bits)
4 (low bits)
7
7
4
9
9
6
rdAdr0
rdAdr1
wrAdr
eftdwe
rdAdr0
rdAdr1
wrAdr
evtdwe
encoded fixed tag data
encoded variable tag data
dataIn 8
out0 1
out1 1
dataIn 8
out0 1
out1 1
advTag
1
0
1
0
1
1
etd0
etd1

FIG. 217

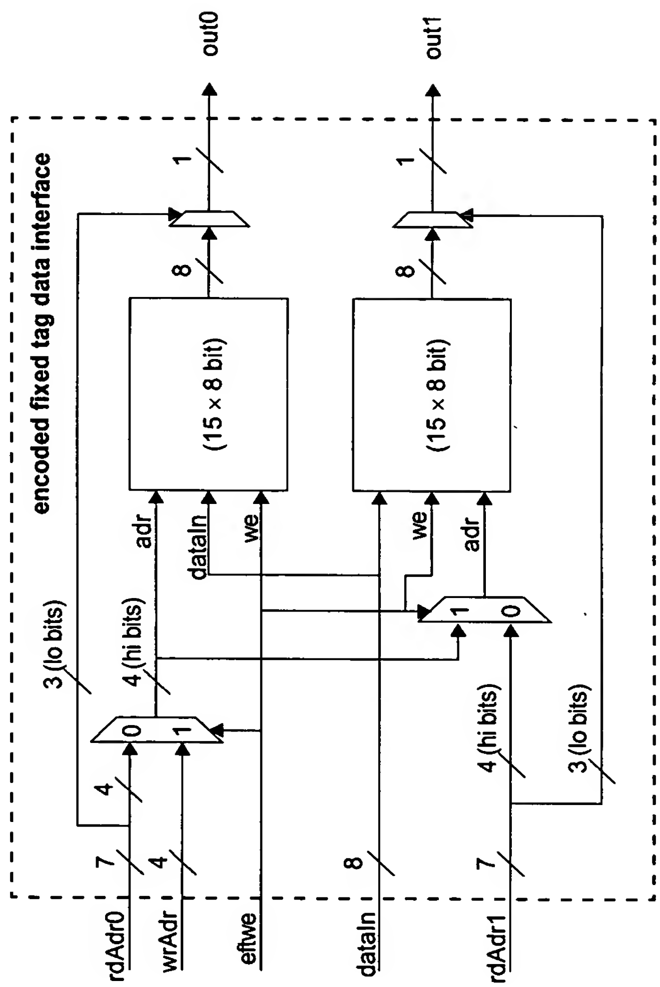

encoded fixed tag data interface
rdAdr0
wrAdr
eftwe
dataIn
rdAdr1
7
4
4
8
7
0
1
3 (lo bits)
4 (hi bits)
4 (hi bits)
3 (lo bits)
adr
dataIn
we
we
adr
(15 × 8 bit)
(15 × 8 bit)
8
8
1
1
out0
out1

FIG. 218

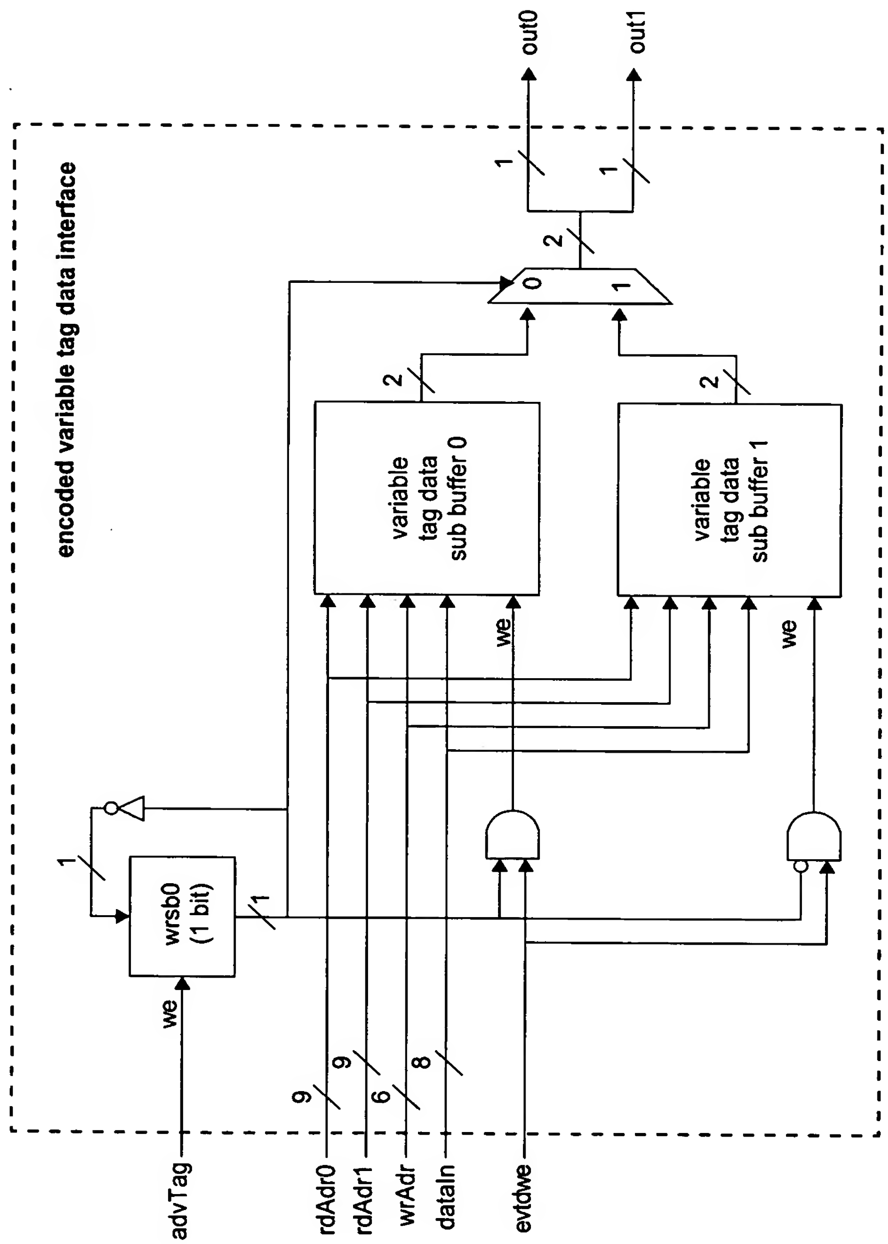

encoded variable tag data interface
advTag
we
wrsb0
(1 bit)
1
1
rdAdr0
rdAdr1
wrAdr
dataIn
evtdwe
9
9
6
8
we
we
variable
tag data
sub buffer 0
variable
tag data
sub buffer 1
2
2
0
1
2
1
1
out0
out1

FIG. 219

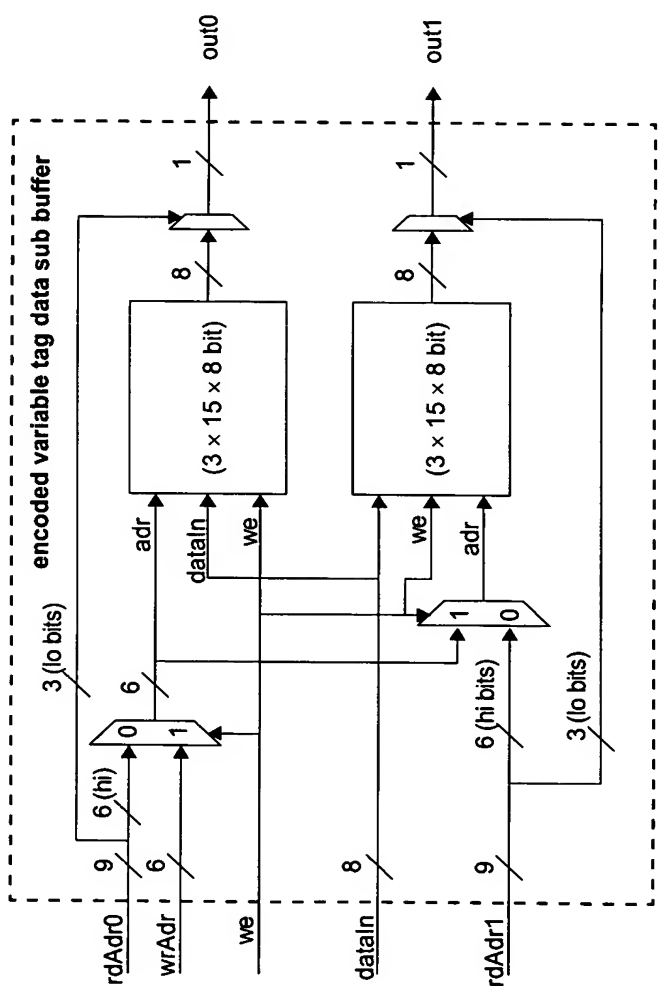
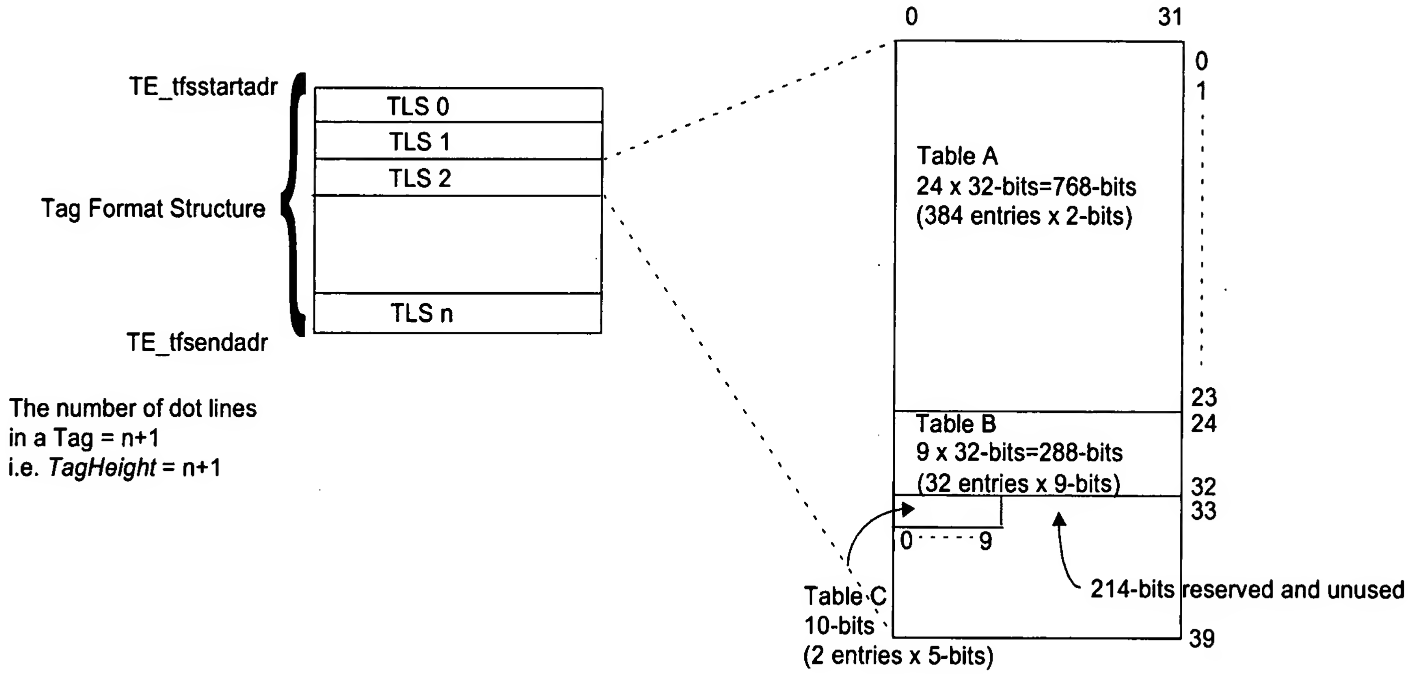

encoded variable tag data sub buffer
(3 × 15 × 8 bit)
(3 × 15 × 8 bit)
adr
dataIn
we
we
adr
out0
out1
rdAdr0
wrAdr
we
dataIn
rdAdr1
3 (lo bits)
6 (hi)
6 (hi bits)
3 (lo bits)

FIG. 220



TE_tfsstartadr
Tag Format Structure
TE_tfsendadr
TLS 0
TLS 1
TLS 2
TLS n
The number of dot lines
in a Tag = n+1
i.e. *TagHeight* = n+1
0
31
Table A
24 x 32-bits=768-bits
(384 entries x 2-bits)
Table B
9 x 32-bits=288-bits
(32 entries x 9-bits)
0 9
Table C
10-bits
(2 entries x 5-bits)
214-bits reserved and unused
0
1
23
24
32
33
39

FIG. 221

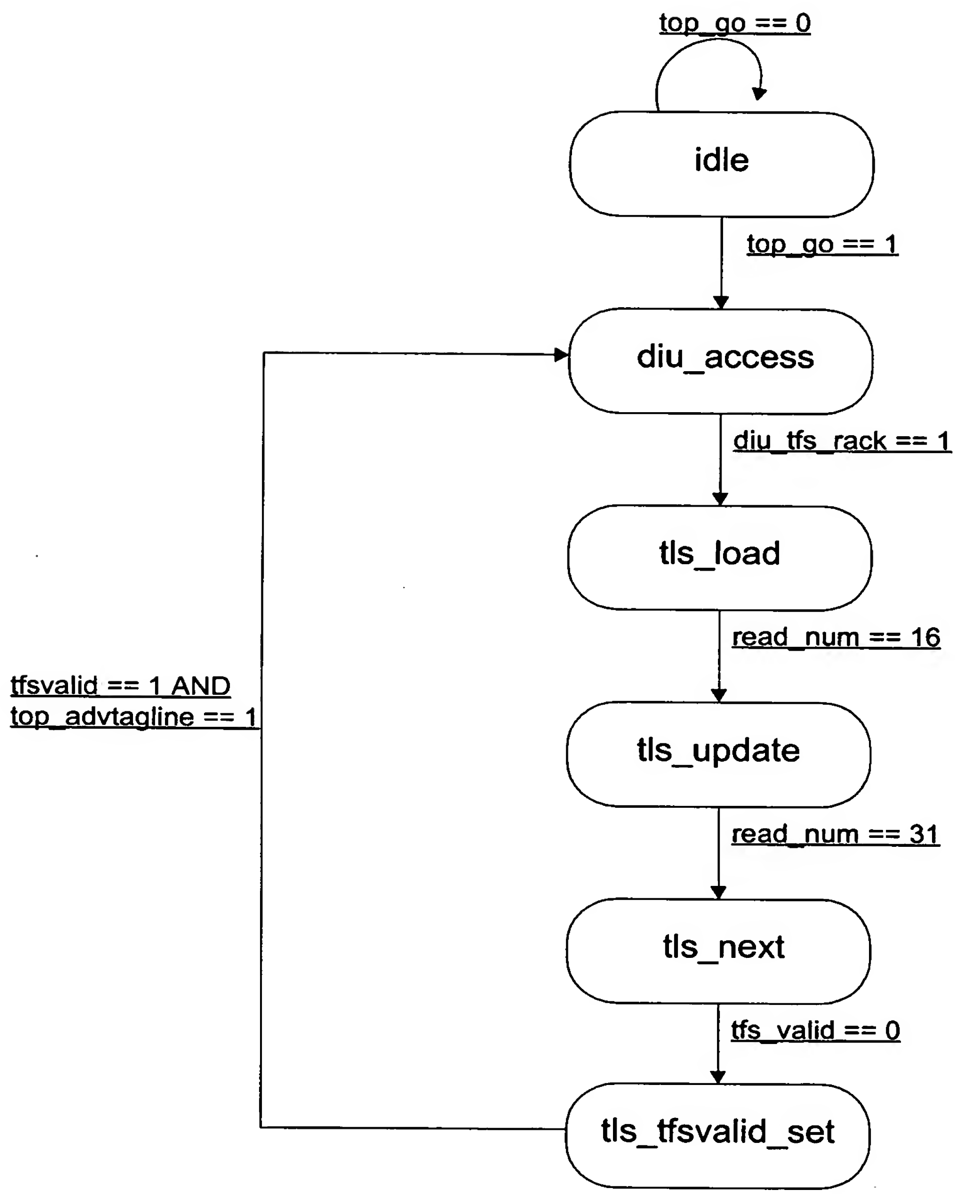

top_go == 0
idle
top_go == 1
diu_access
diu_tfs_rack == 1
tls_load
read_num == 16
tls_update
read_num == 31
tls_next
tfs_valid == 0
tls_tfsvalid_set
tfsvalid == 1 AND
top_advtagline == 1

FIG. 222

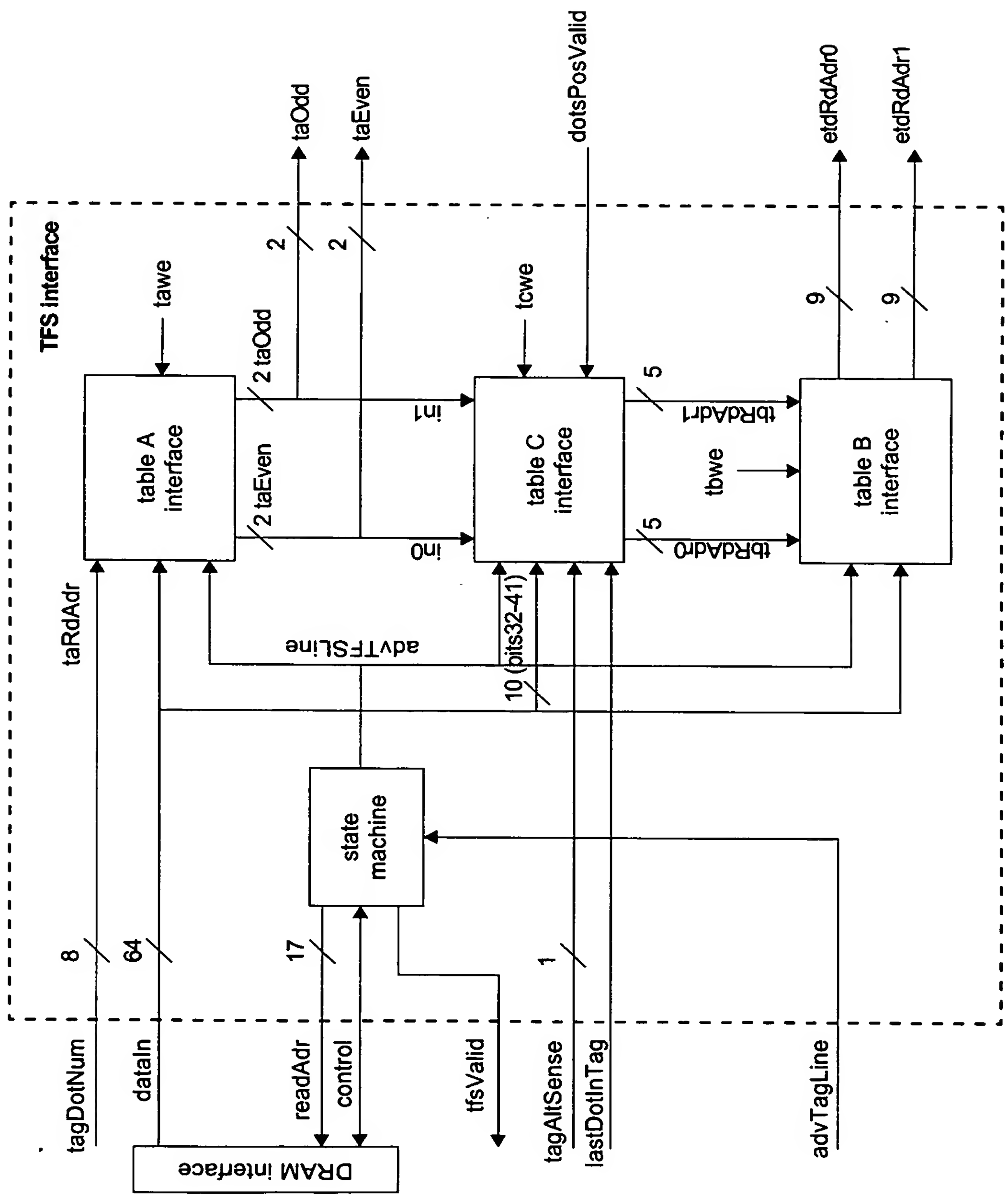

TFS Interface
table A interface
table B interface
table C interface
state machine
DRAM interface
tawe
tcwe
tbwe
taOdd
taEven
2 taOdd
2 taEven
2
2
dotsPosValid
etdRdAdr0
etdRdAdr1
9
9
5
5
in1
in0
tbRdAdr1
tbRdAdr0
taRdAdr
advTFSLine
10 (bits32-41)
8
64
17
1
tagDotNum
dataIn
readAdr
control
tfsValid
tagAltSense
lastDotInTag
advTagLine

FIG. 223

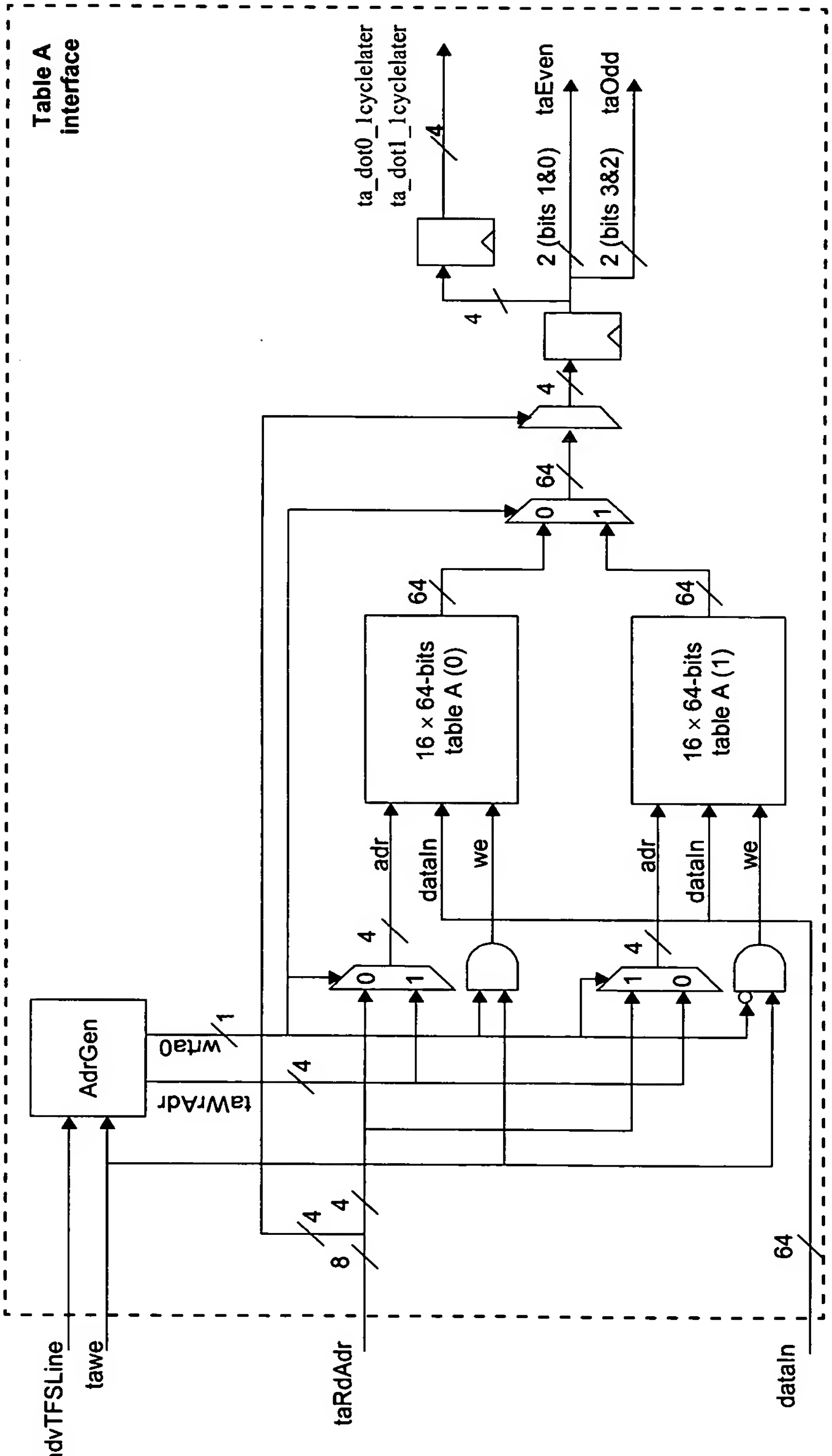


Table A interface
ta_dot0_1cyclelater
ta_dot1_1cyclelater
taEven
taOdd
2 (bits 1&0)
2 (bits 3&2)
4
64
16 × 64-bits table A (0)
16 × 64-bits table A (1)
adr
dataIn
we
AdrGen
wrta0
taWrAdr
1
8
advTFSLine
tawe
taRdAdr
dataIn

FIG. 224

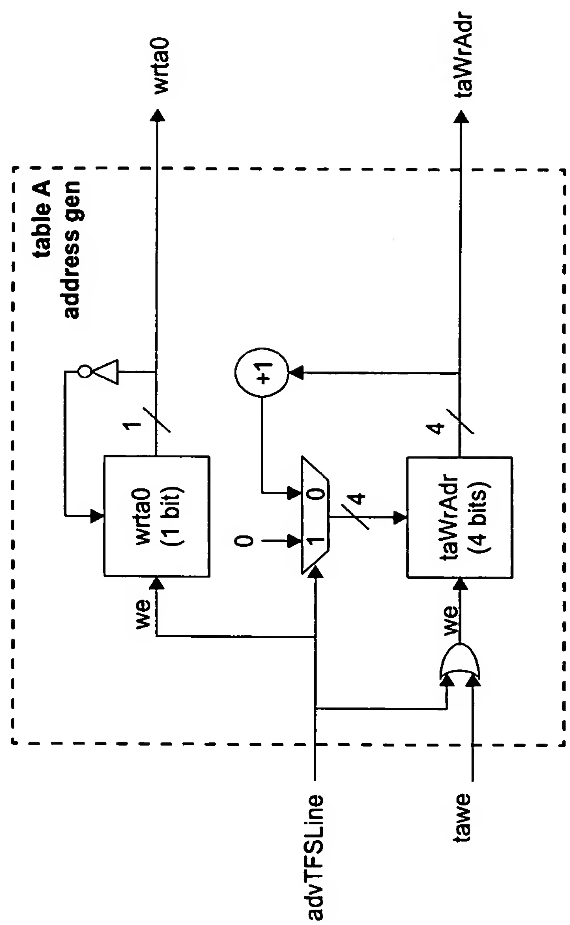


table A
address gen
wrta0
taWrAdr
wrta0
(1 bit)
taWrAdr
(4 bits)
+1
we
we
1
4
4
0
1
0
advTFSLine
tawe

FIG. 225

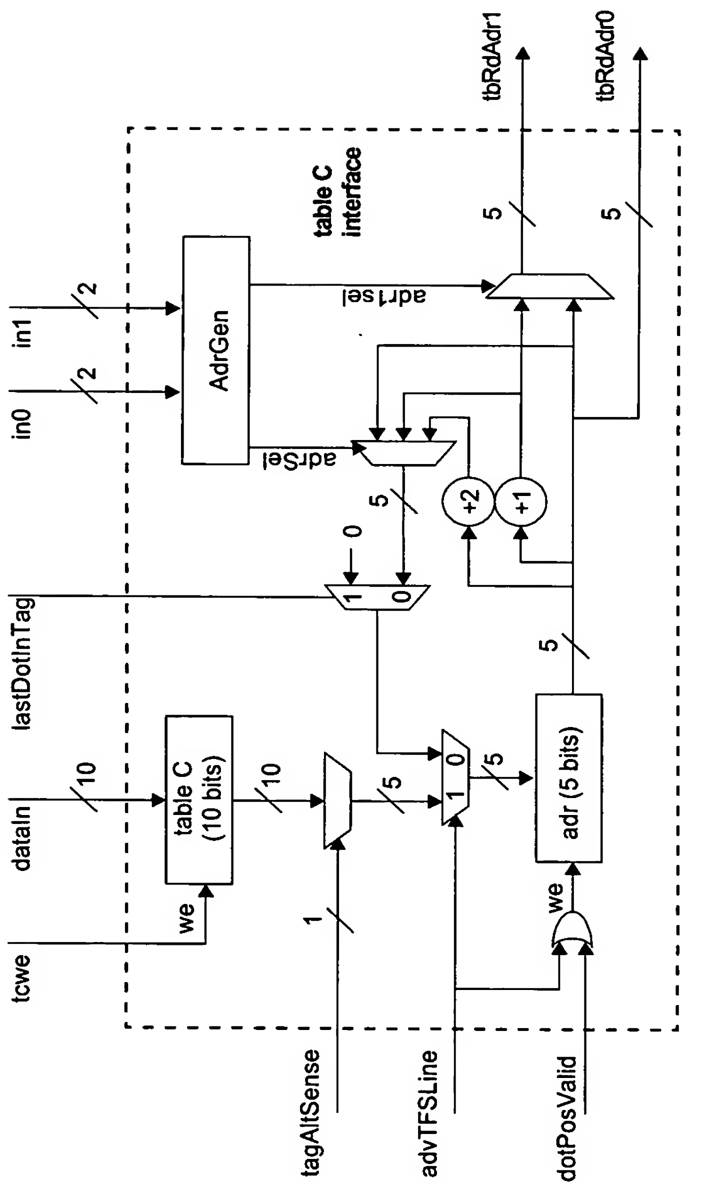
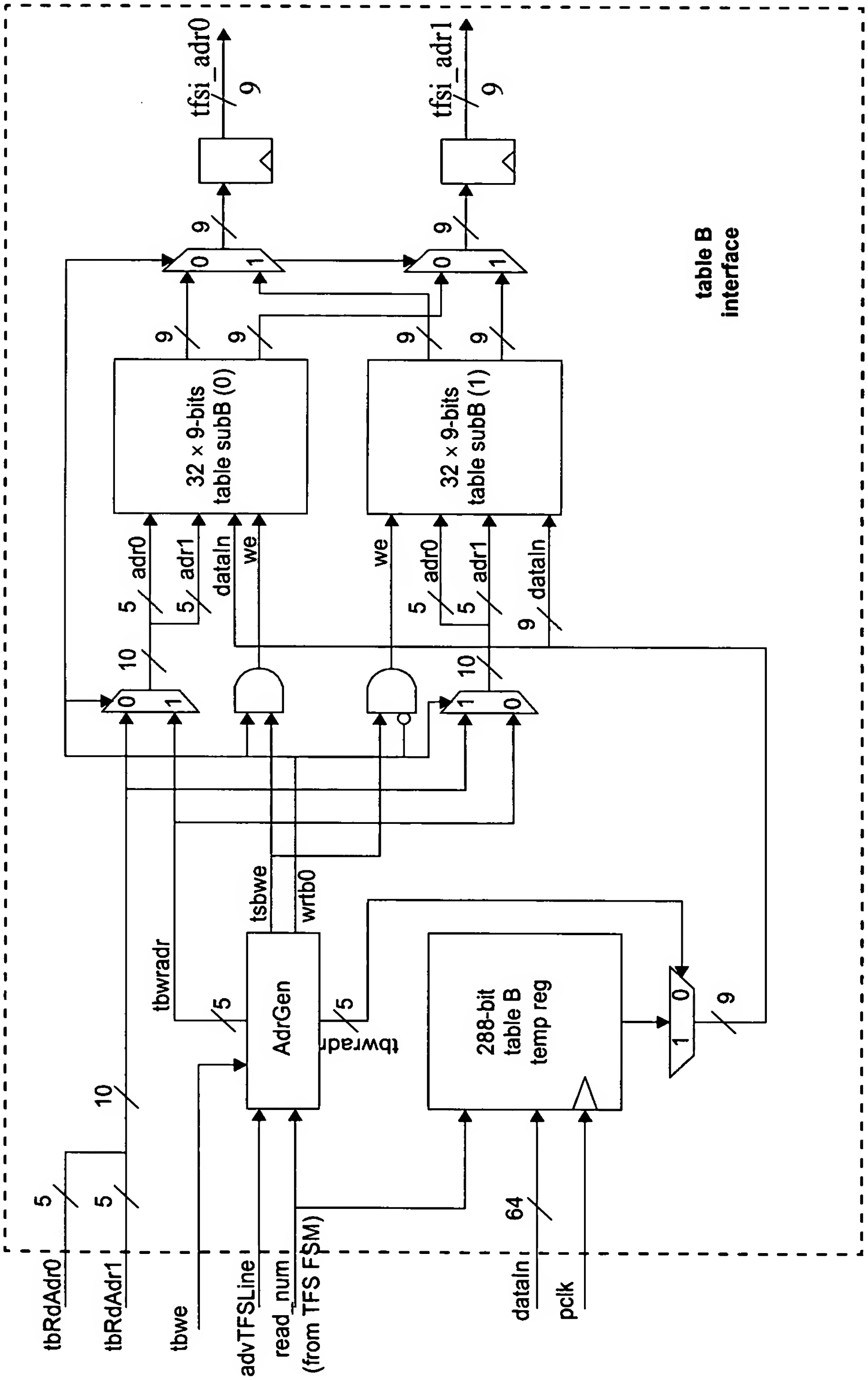

tcwe
dataIn
lastDotInTag
in0
in1
we
table C (10 bits)
AdrGen
adrSel
adr1sel
table C interface
tbRdAdr1
tbRdAdr0
adr (5 bits)
+2
+1
tagAltSense
advTFSLine
dotPosValid
10
10
5
5
5
5
5
5
2
2
1
0
1
0
1
0
we

FIG. 226



tfsi_adr0
tfsi_adr1
9
table B interface
32 × 9-bits table subB (0)
32 × 9-bits table subB (1)
adr0
adr1
dataIn
we
5
10
tbwradr
tsbwe
wrtb0
AdrGen
288-bit table B temp reg
64
tbRdAdr0
tbRdAdr1
tbwe
advTFSLine
read_num (from TFS FSM)
dataIn
pclk

FIG. 227

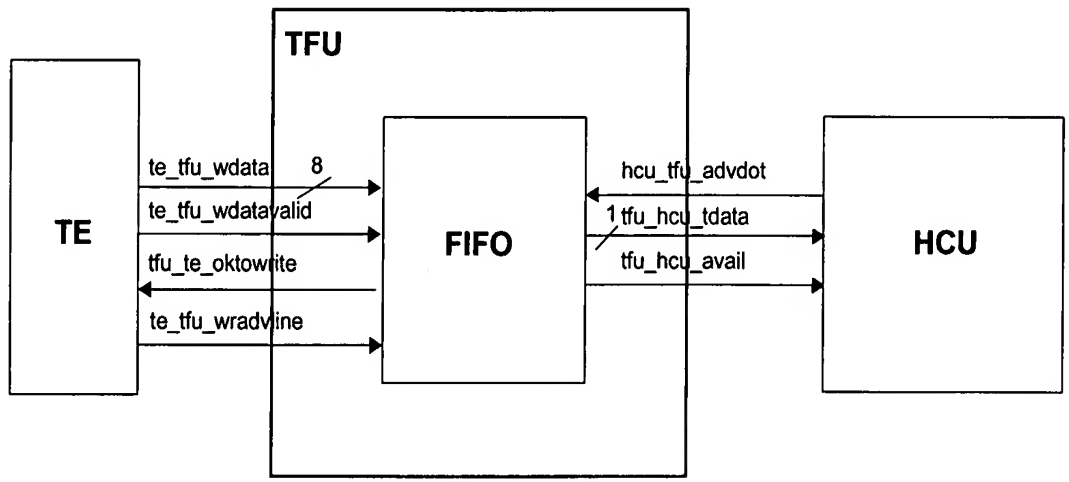


FIG. 228

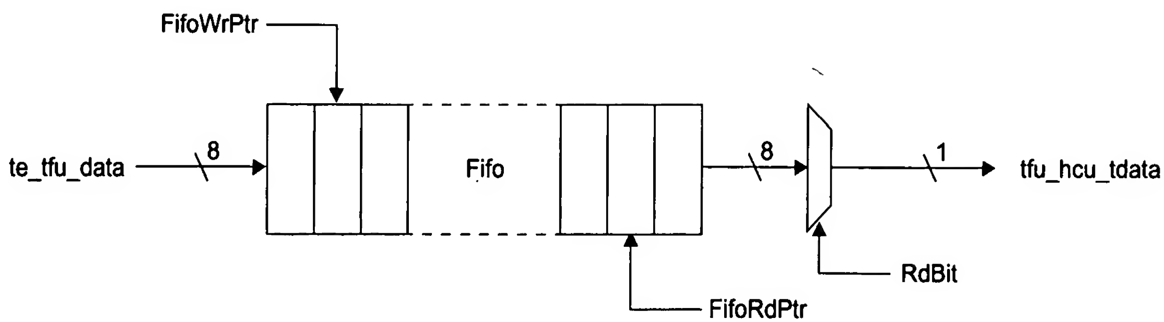


FIG. 229

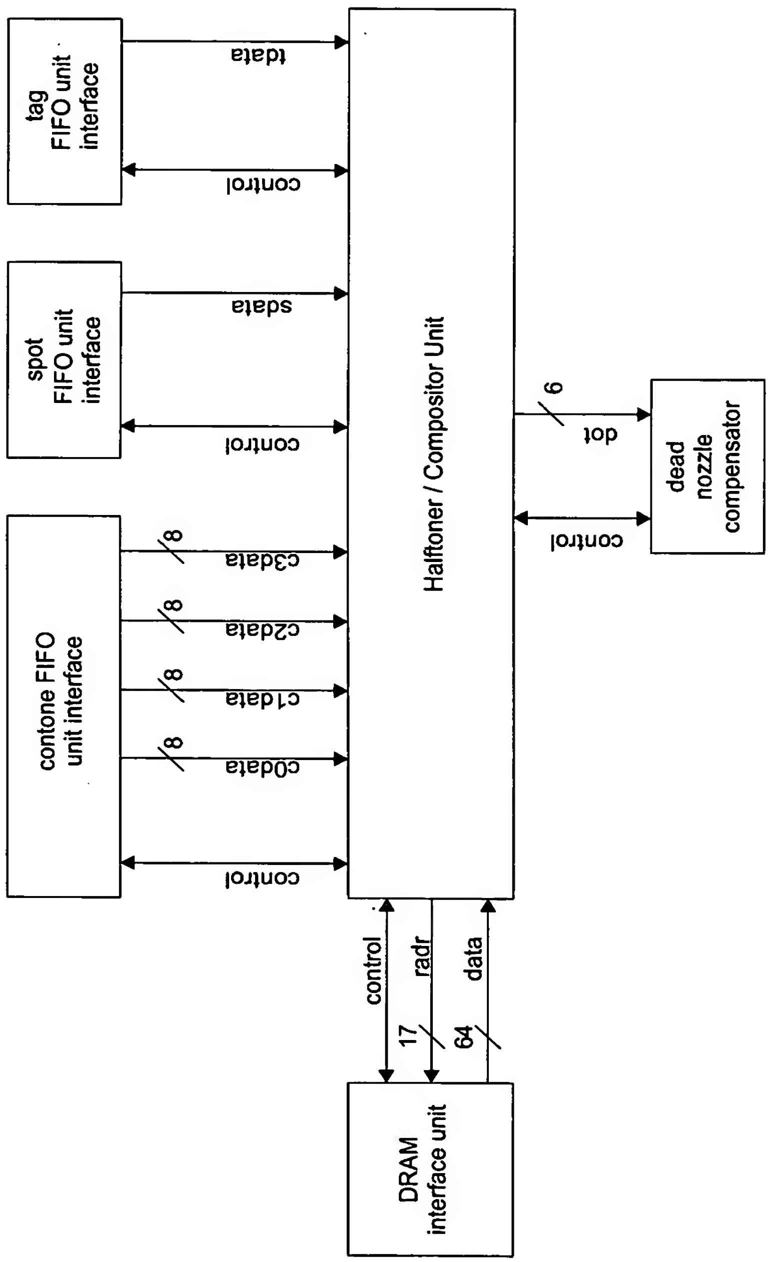

tag FIFO unit interface
spot FIFO unit interface
contone FIFO unit interface
tdata
control
sdata
control
c3data
c2data
c1data
c0data
control
8
8
8
8
Halftoner / Compositor Unit
6
dot
control
dead nozzle compensator
control
radr
data
17
64
DRAM interface unit

FIG. 230

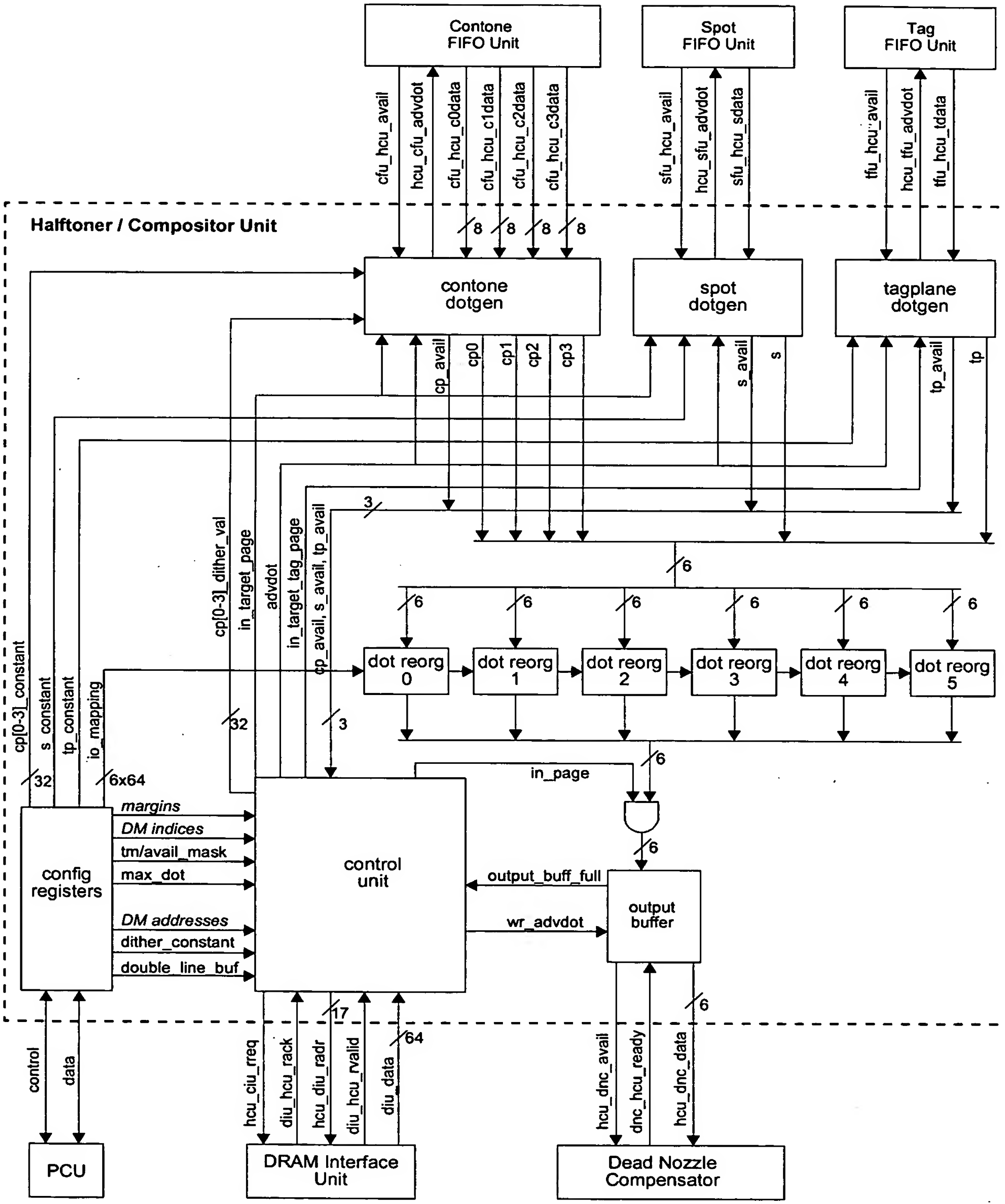
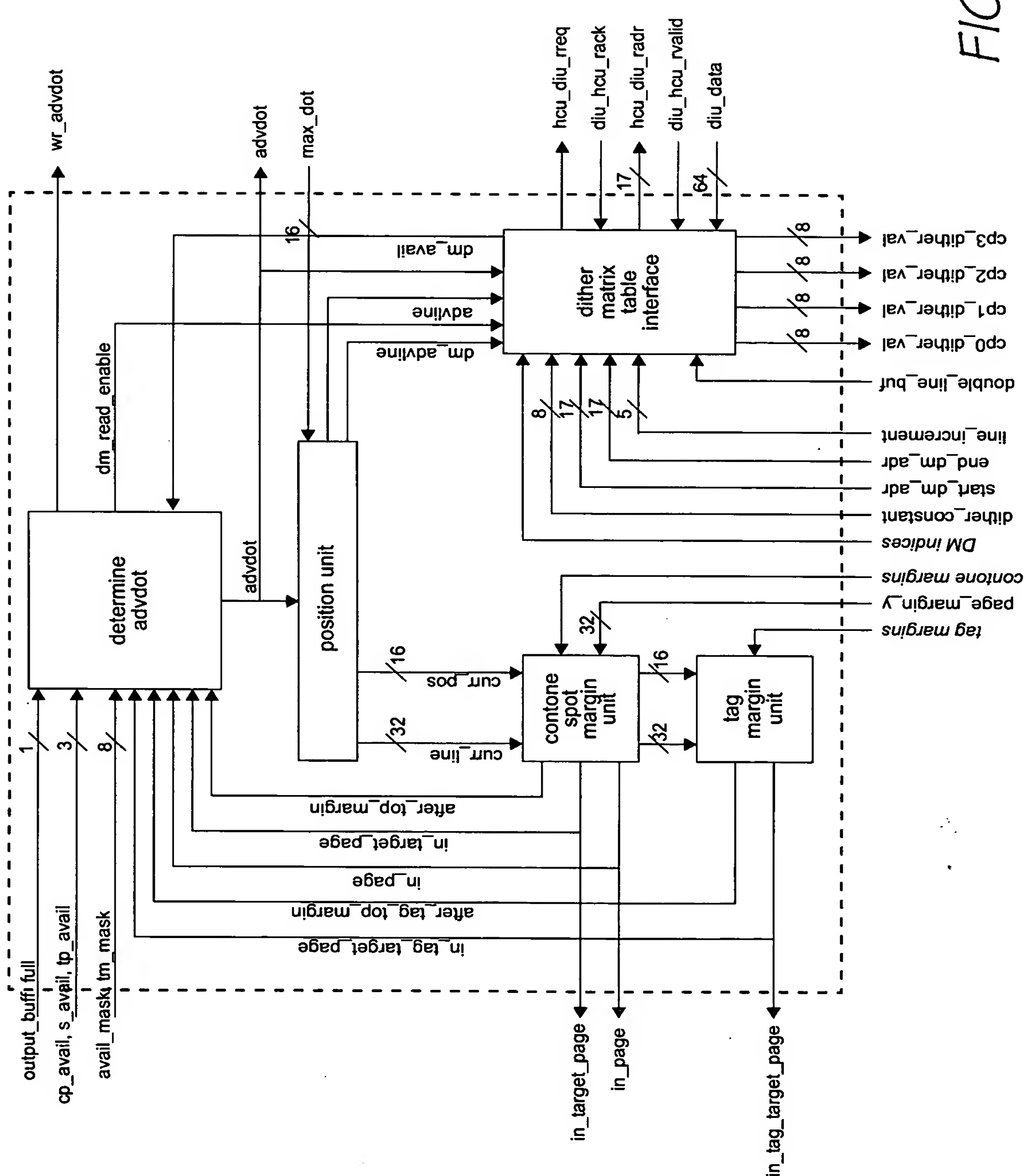

Contone FIFO Unit
Spot FIFO Unit
Tag FIFO Unit
cfu_hcu_avail
hcu_cfu_advdot
cfu_hcu_c0data
cfu_hcu_c1data
cfu_hcu_c2data
cfu_hcu_c3data
sfu_hcu_avail
hcu_sfu_advdot
sfu_hcu_sdata
tfu_hcu_avail
hcu_tfu_advdot
tfu_hcu_tdata
Halftoner / Compositor Unit
8
contone dotgen
spot dotgen
tagplane dotgen
cp_avail
cp0
cp1
cp2
cp3
s_avail
s
tp_avail
tp
3
6
dot reorg 0
dot reorg 1
dot reorg 2
dot reorg 3
dot reorg 4
dot reorg 5
cp[0-3]_constant
s_constant
tp_constant
io_mapping
cp[0-3]_dither_val
in_target_page
advdot
in_target_tag_page
cp_avail, s_avail, tp_avail
32
6x64
in_page
margins
DM indices
tm/avail_mask
max_dot
DM addresses
dither_constant
double_line_buf
config registers
control unit
output_buff_full
wr_advdot
output buffer
17
64
control
data
hcu_ciu_rreq
diu_hcu_rack
hcu_diu_radr
diu_hcu_rvalid
diu_data
hcu_dnc_avail
dnc_hcu_ready
hcu_dnc_data
PCU
DRAM Interface Unit
Dead Nozzle Compensator

FIG. 231



wr_advdot
advdot
max_dot
hcu_diu_rreq
diu_hcu_rack
hcu_diu_radr
diu_hcu_rvalid
diu_data
17
64
16
dm_avail
advline
dm_advline
dm_read_enable
dither matrix table interface
8
cp3_dither_val
cp2_dither_val
cp1_dither_val
cp0_dither_val
double_line_buf
line_increment
end_dm_adr
start_dm_adr
dither_constant
DM indices
17
5
contone margins
page_margin_y
tag margins
determine advdot
advdot
position unit
curr_pos
curr_line
32
contone spot margin unit
tag margin unit
after_top_margin
in_target_page
in_page
after_tag_top_margin
in_tag_target_page
1
3
output_buff full
cp_avail, s_avail, tp_avail
avail_mask, tm_mask
in_target_page
in_page
in_tag_target_page

FIG. 232

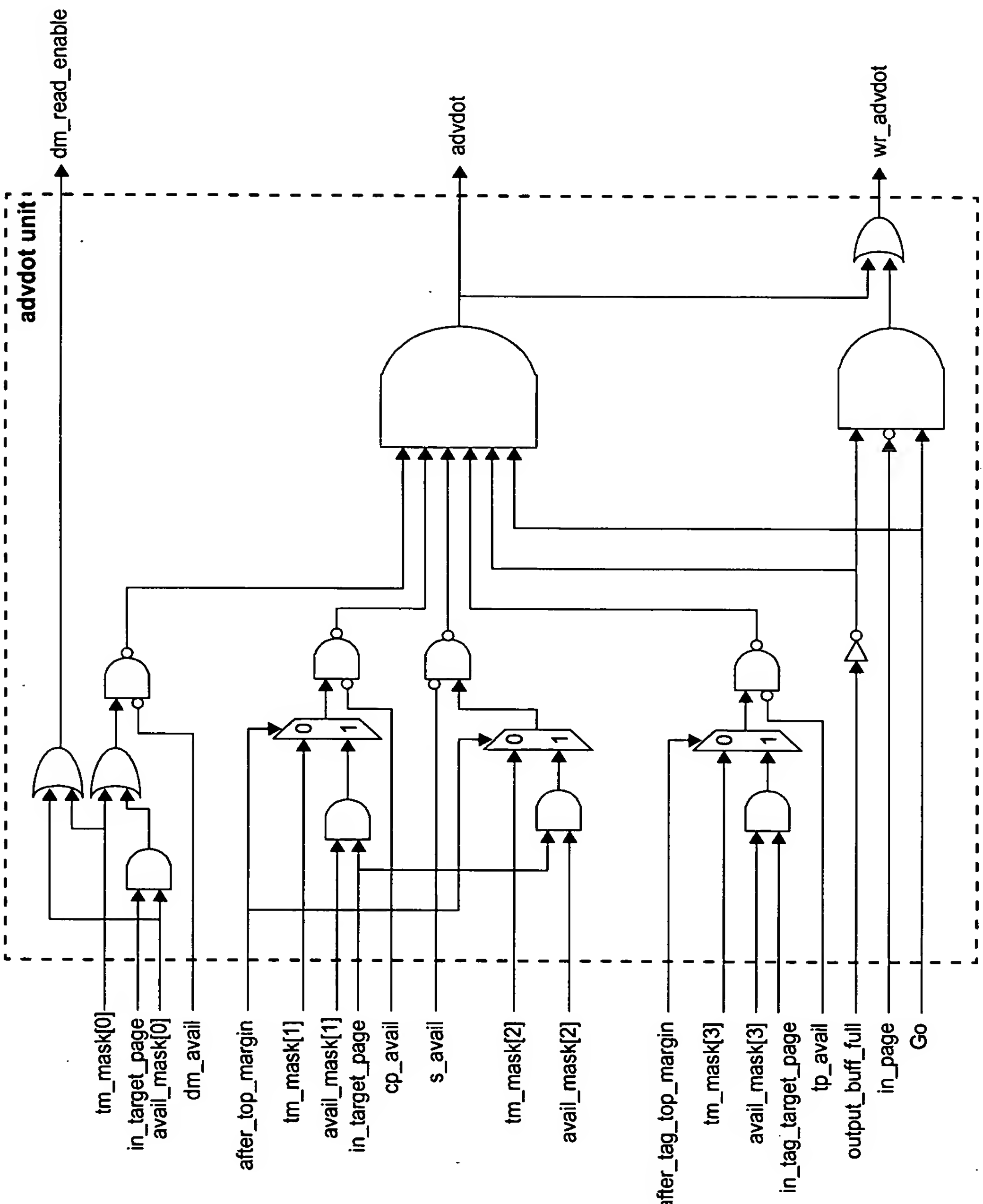

advdot unit
dm_read_enable
advdot
wr_advdot
0
1
0
1
0
1
tm_mask[0]
in_target_page
avail_mask[0]
dm_avail
after_top_margin
tm_mask[1]
avail_mask[1]
in_target_page
cp_avail
s_avail
tm_mask[2]
avail_mask[2]
after_tag_top_margin
tm_mask[3]
avail_mask[3]
in_tag_target_page
tp_avail
output_buff_full
in_page
Go

FIG. 233

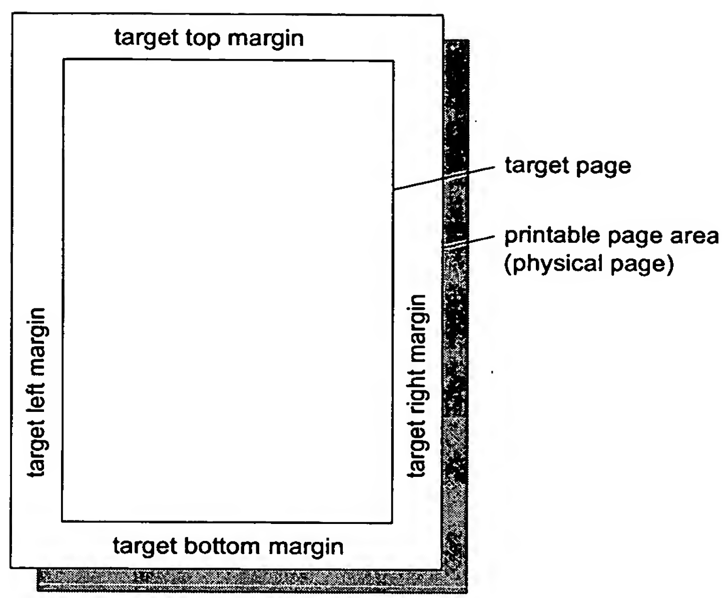

target top margin
target page
printable page area
(physical page)
target left margin
target right margin
target bottom margin

FIG. 234

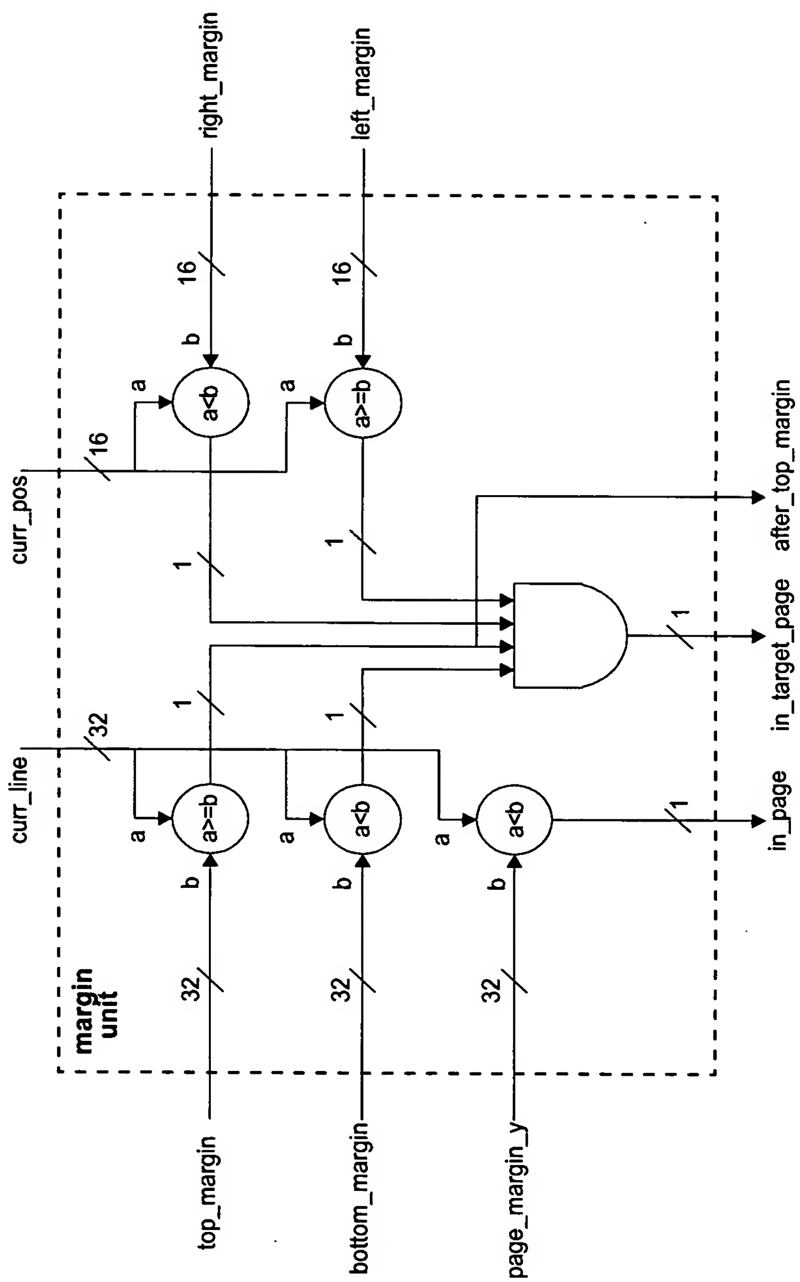
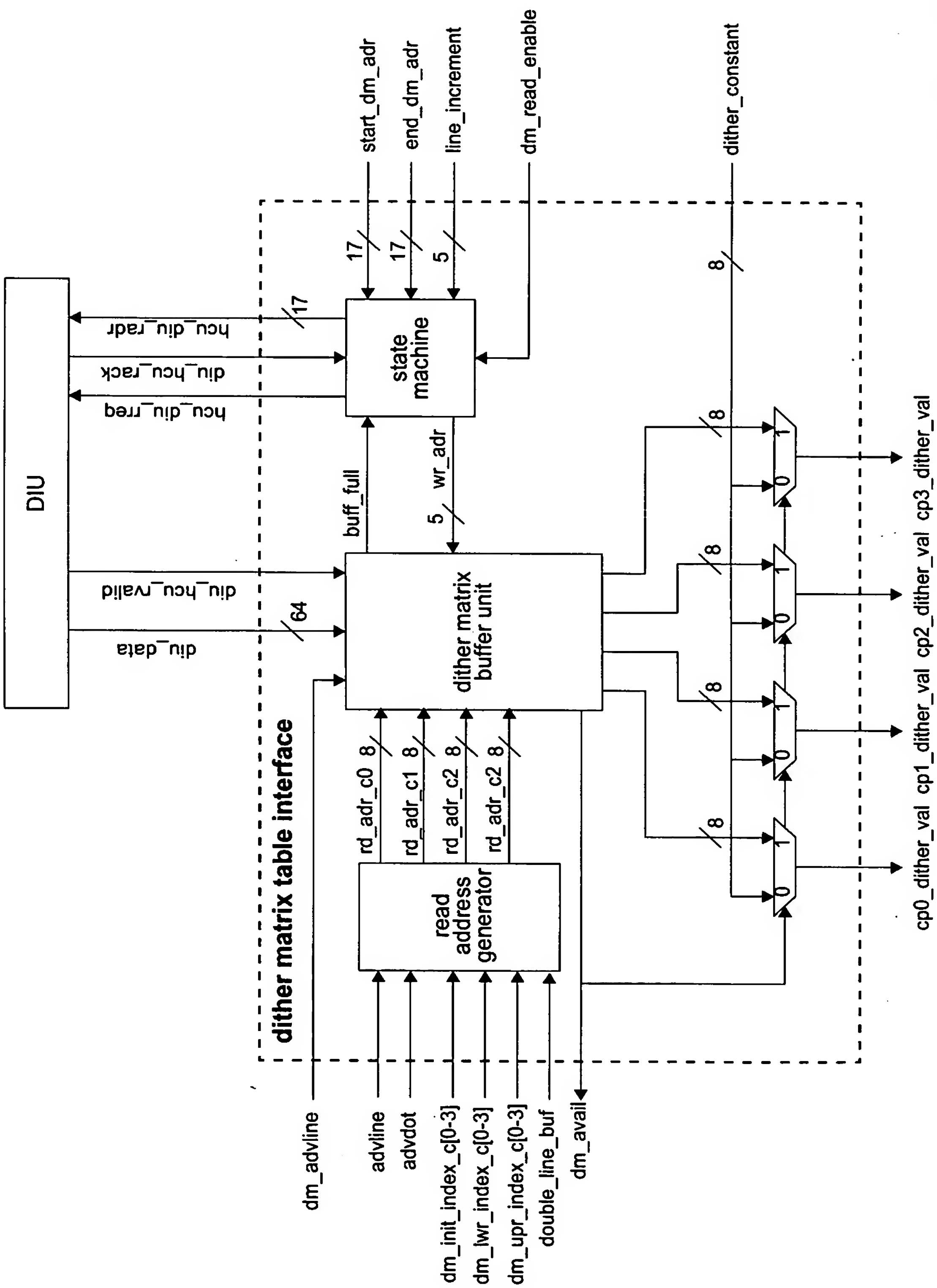

margin unit
top_margin
bottom_margin
page_margin_y
curr_line
curr_pos
right_margin
left_margin
32
32
32
32
16
16
16
1
1
1
1
1
1
a
b
a>=b
a<b
a<b
a<b
a>=b
in_page
in_target_page
after_top_margin

FIG. 235



dither matrix table interface
DIU
state machine
dither matrix buffer unit
read address generator
start_dm_adr
end_dm_adr
line_increment
dm_read_enable
dither_constant
hcu_diu_radr
diu_hcu_rack
hcu_diu_rreq
diu_hcu_rvalid
diu_data
buff_full
wr_adr
17
17
17
5
5
8
64
rd_adr_c0
rd_adr_c1
rd_adr_c2
rd_adr_c2
8
8
8
8
dm_advline
advline
advdot
dm_init_index_c[0-3]
dm_lwr_index_c[0-3]
dm_upr_index_c[0-3]
double_line_buf
dm_avail
cp0_dither_val
cp1_dither_val
cp2_dither_val
cp3_dither_val
0
1

FIG. 236

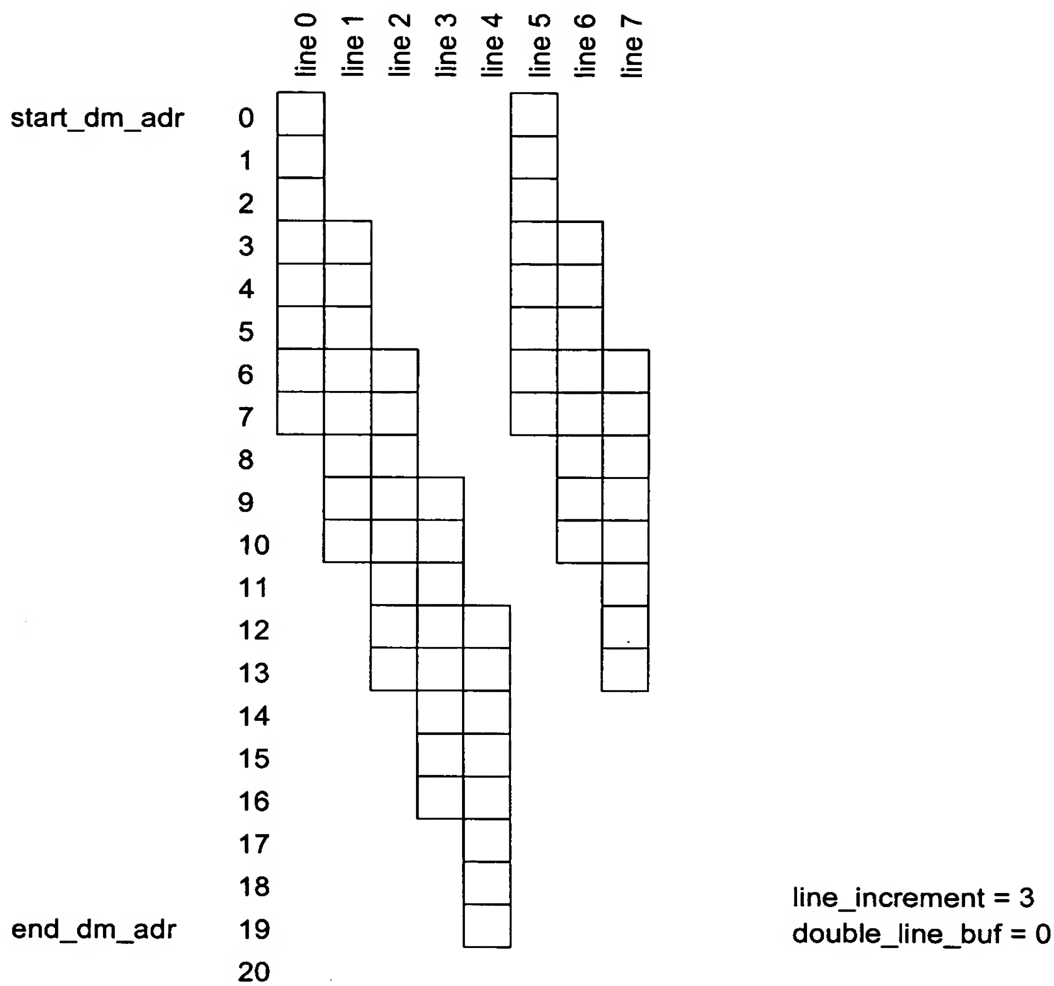

line 0
line 1
line 2
line 3
line 4
line 5
line 6
line 7
start_dm_adr
end_dm_adr
0
1
2
3
4
5
6
7
8
9
10
11
12
13
14
15
16
17
18
19
20
line_increment = 3
double_line_buf = 0

FIG. 237

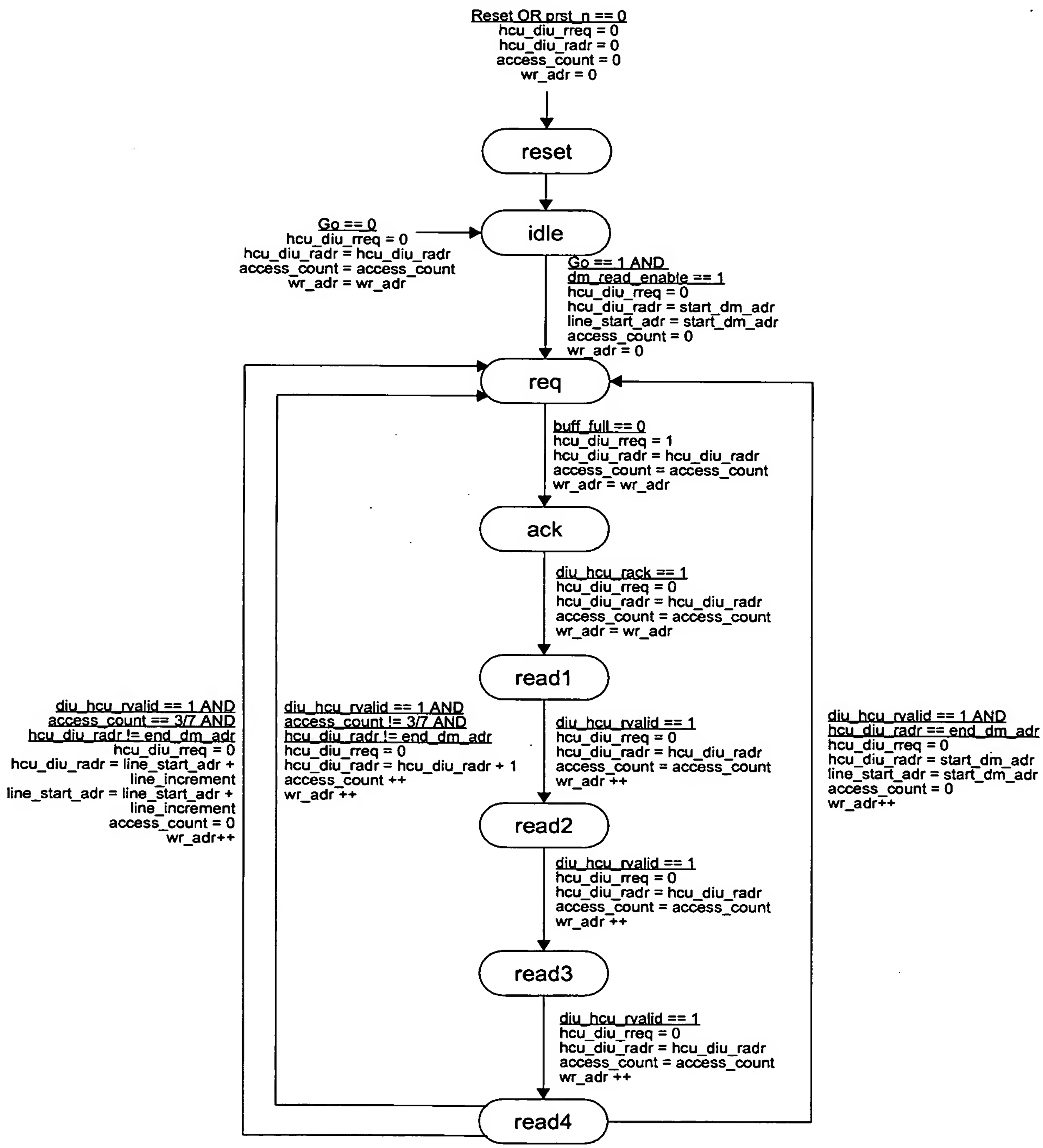

Reset OR prst_n == 0
hcu_diu_rreq = 0
hcu_diu_radr = 0
access_count = 0
wr_adr = 0
reset
idle
Go == 0
hcu_diu_rreq = 0
hcu_diu_radr = hcu_diu_radr
access_count = access_count
wr_adr = wr_adr
Go == 1 AND
dm_read_enable == 1
hcu_diu_rreq = 0
hcu_diu_radr = start_dm_adr
line_start_adr = start_dm_adr
access_count = 0
wr_adr = 0
req
buff_full == 0
hcu_diu_rreq = 1
hcu_diu_radr = hcu_diu_radr
access_count = access_count
wr_adr = wr_adr
ack
diu_hcu_rack == 1
hcu_diu_rreq = 0
hcu_diu_radr = hcu_diu_radr
access_count = access_count
wr_adr = wr_adr
read1
diu_hcu_rvalid == 1
hcu_diu_rreq = 0
hcu_diu_radr = hcu_diu_radr
access_count = access_count
wr_adr ++
read2
diu_hcu_rvalid == 1
hcu_diu_rreq = 0
hcu_diu_radr = hcu_diu_radr
access_count = access_count
wr_adr ++
read3
diu_hcu_rvalid == 1
hcu_diu_rreq = 0
hcu_diu_radr = hcu_diu_radr
access_count = access_count
wr_adr ++
read4
diu_hcu_rvalid == 1 AND
access_count == 3/7 AND
hcu_diu_radr != end_dm_adr
hcu_diu_rreq = 0
hcu_diu_radr = line_start_adr +
line_increment
line_start_adr = line_start_adr +
line_increment
access_count = 0
wr_adr++
diu_hcu_rvalid == 1 AND
access_count != 3/7 AND
hcu_diu_radr != end_dm_adr
hcu_diu_rreq = 0
hcu_diu_radr = hcu_diu_radr + 1
access_count ++
wr_adr ++
diu_hcu_rvalid == 1 AND
hcu_diu_radr == end_dm_adr
hcu_diu_rreq = 0
hcu_diu_radr = start_dm_adr
line_start_adr = start_dm_adr
access_count = 0
wr_adr++

FIG. 238

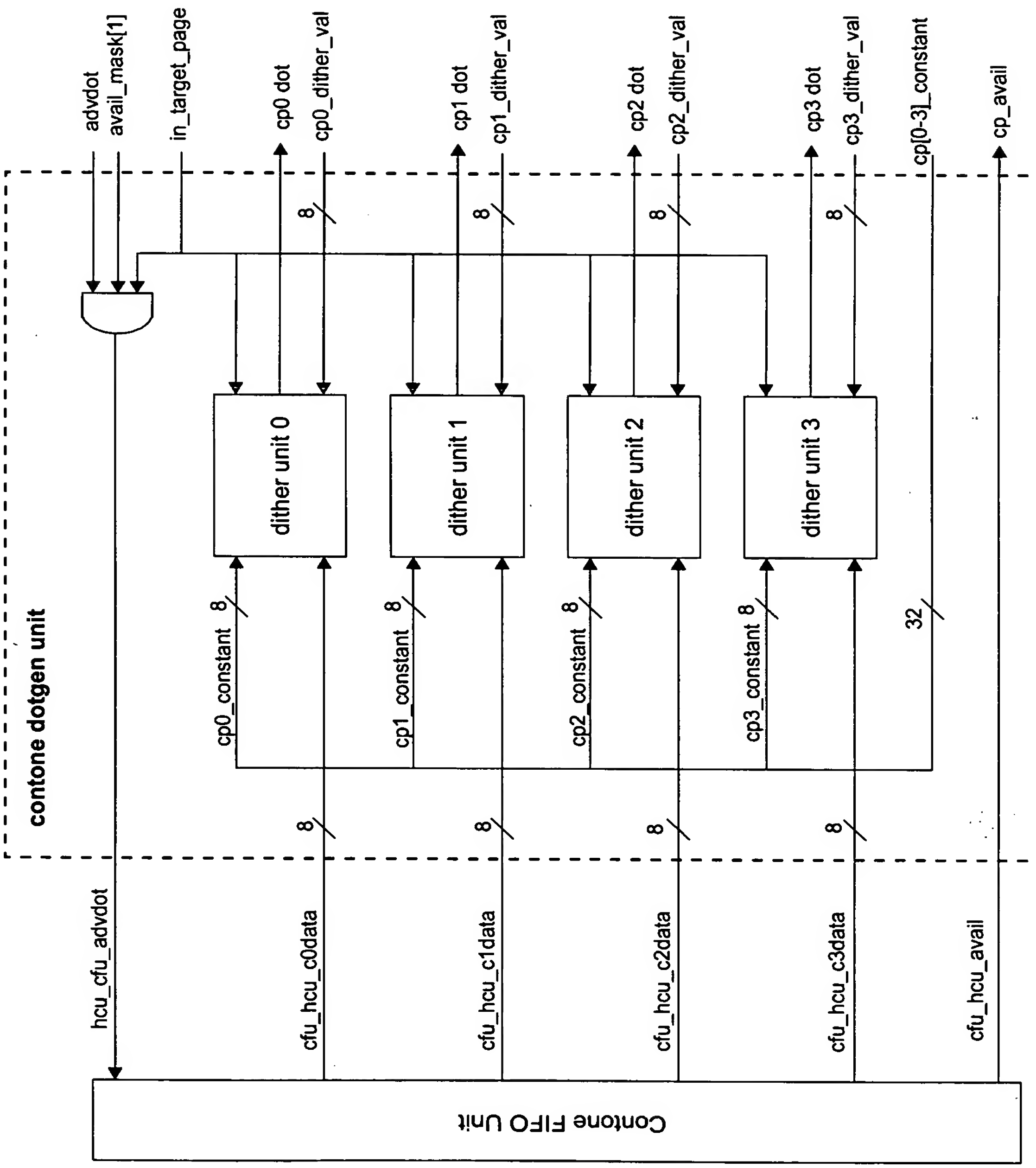

contone dotgen unit
advdot
avail_mask[1]
in_target_page
cp0 dot
cp0_dither_val
cp1 dot
cp1_dither_val
cp2 dot
cp2_dither_val
cp3 dot
cp3_dither_val
cp[0-3]_constant
cp_avail
8
dither unit 0
dither unit 1
dither unit 2
dither unit 3
cp0_constant
cp1_constant
cp2_constant
cp3_constant
32
hcu_cfu_advdot
cfu_hcu_c0data
cfu_hcu_c1data
cfu_hcu_c2data
cfu_hcu_c3data
cfu_hcu_avail
Contone FIFO Unit

FIG. 239

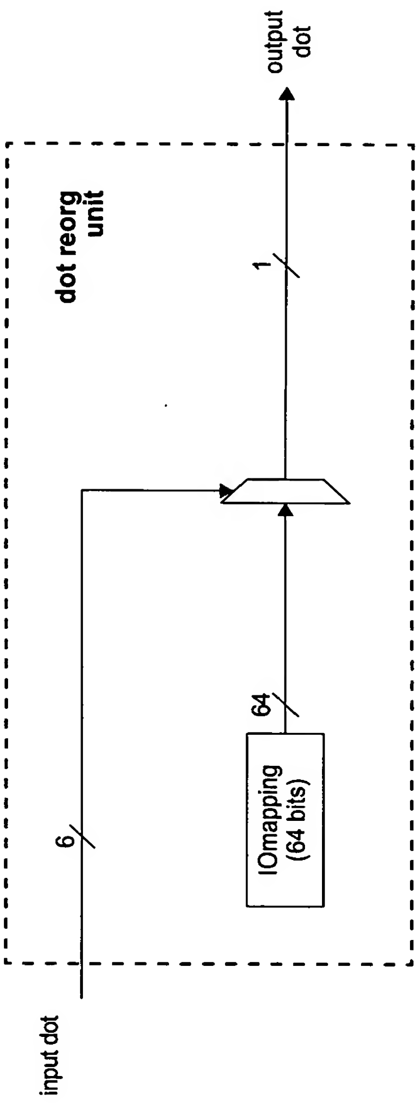


FIG. 240

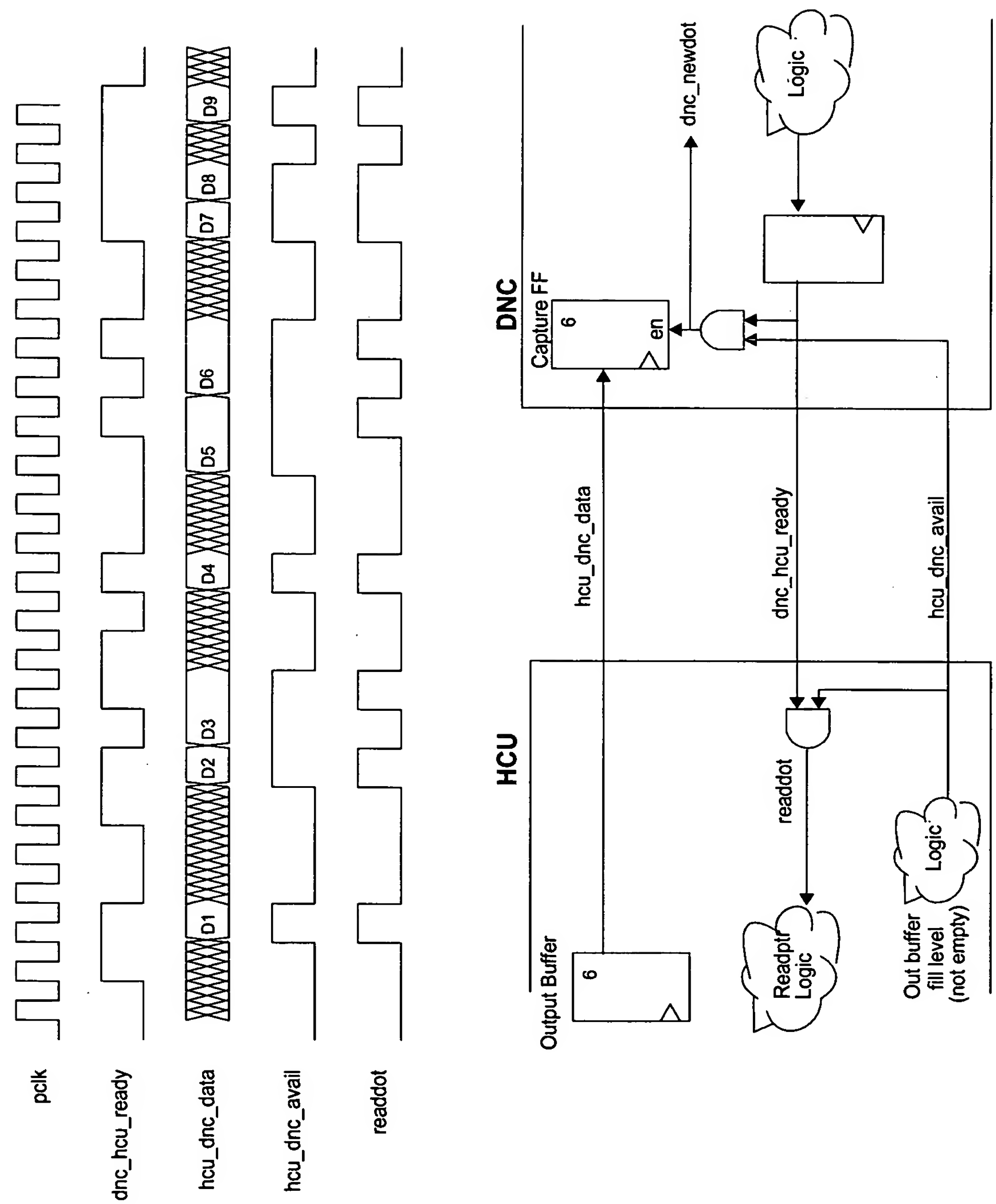

pclk
dnc_hcu_ready
hcu_dnc_data
hcu_dnc_avail
readdot
D1
D2
D3
D4
D5
D6
D7
D8
D9
HCU
DNC
Output Buffer
6
Readptr Logic
Logic
Out buffer fill level (not empty)
readdot
hcu_dnc_data
dnc_hcu_ready
hcu_dnc_avail
Capture FF
6
en
dnc_newdot
Logic

FIG. 241

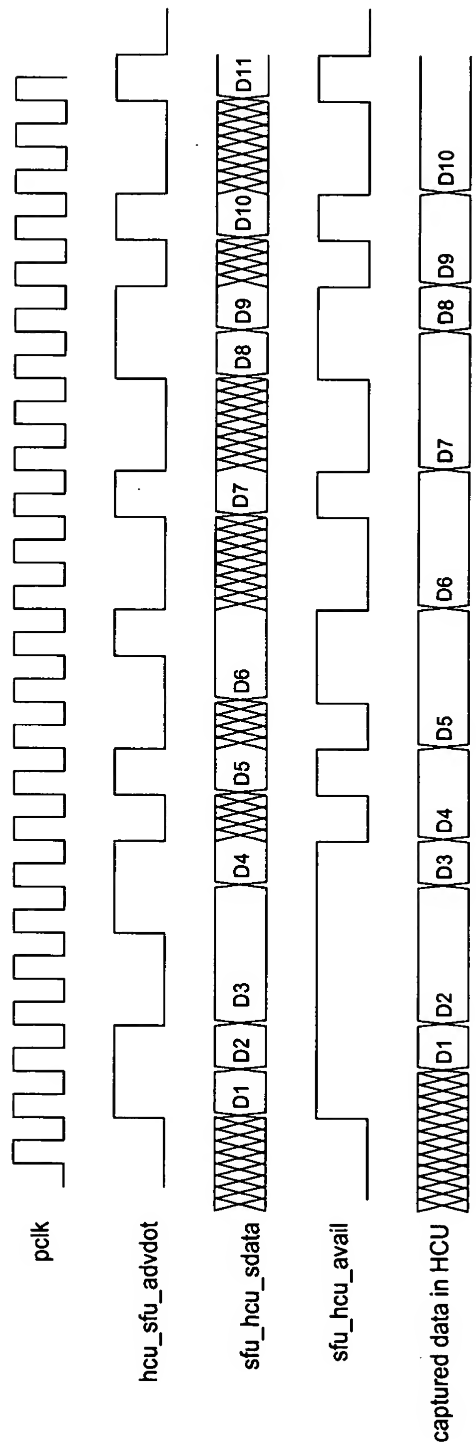

pclk
hcu_sfu_advdot
sfu_hcu_sdata
sfu_hcu_avail
captured data in HCU
D1
D2
D3
D4
D5
D6
D7
D8
D9
D10
D11

FIG. 242

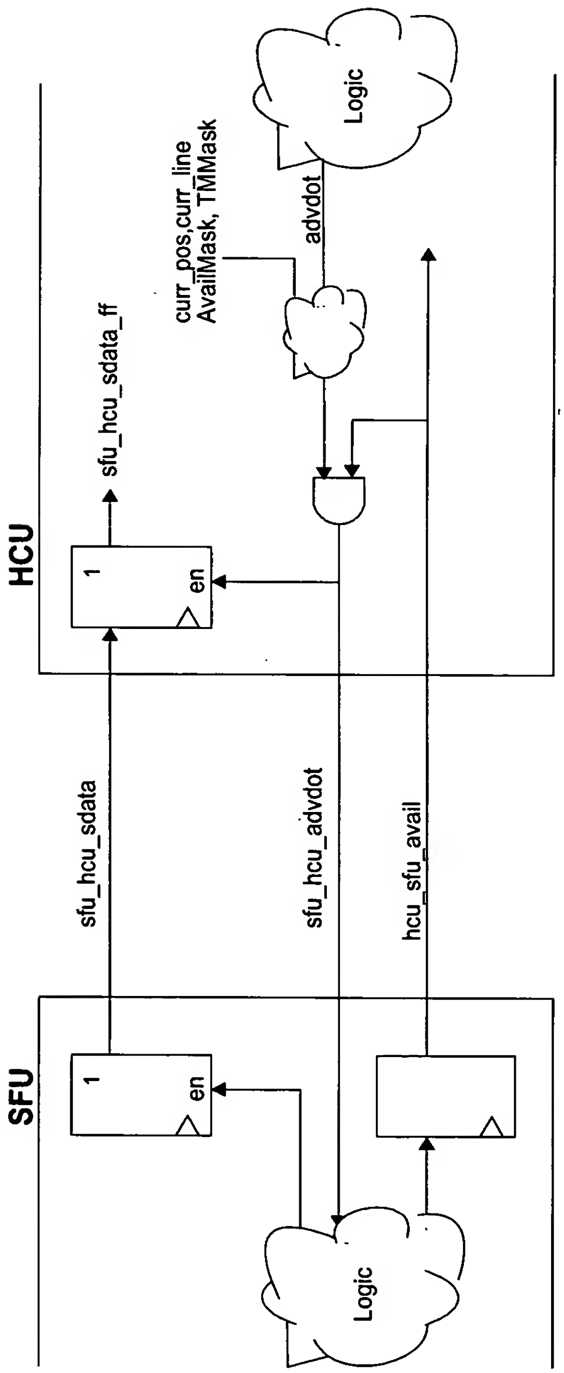

HCU
SFU
Logic
curr_pos,curr_line
AvailMask, TMMask
advdot
sfu_hcu_sdata_ff
1
en
sfu_hcu_sdata
sfu_hcu_advdot
hcu_sfu_avail
1
en
Logic

FIG. 243

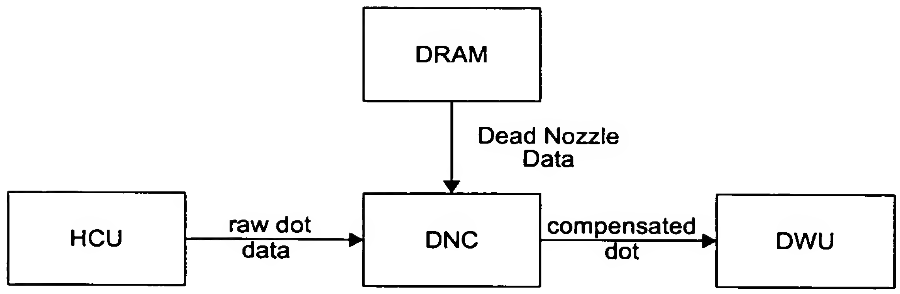


FIG. 244

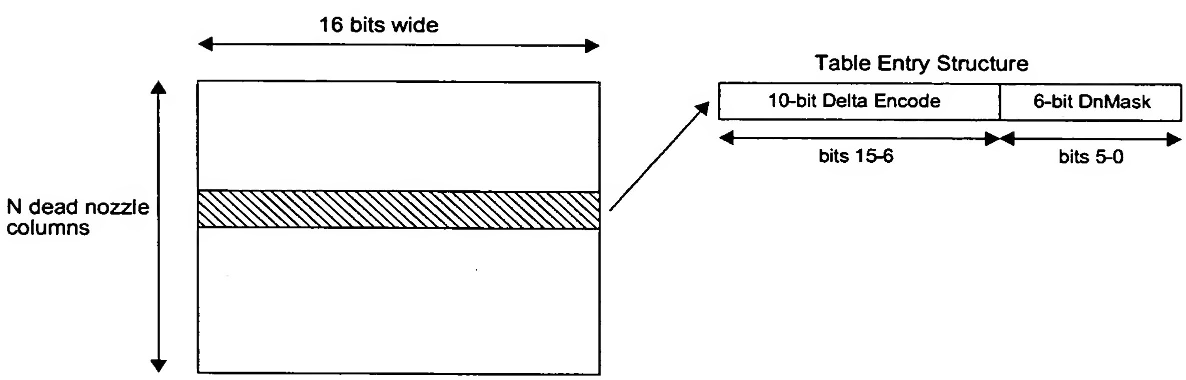


FIG. 245

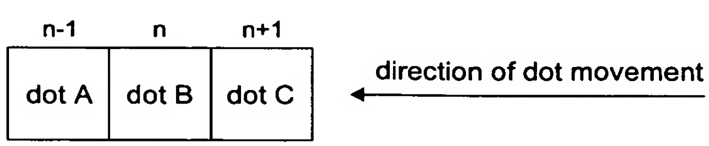


FIG. 246

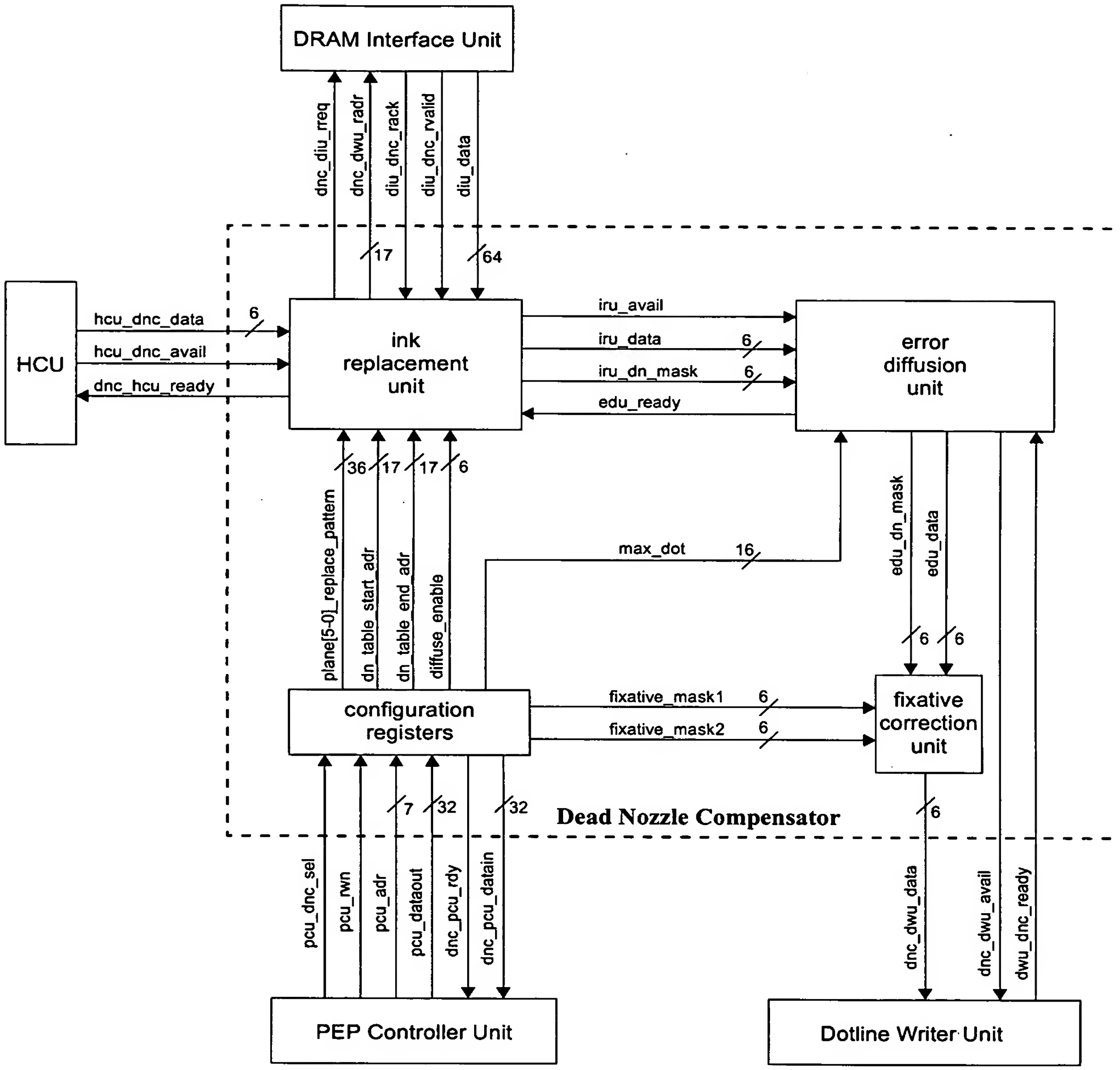

DRAM Interface Unit
dnc_diu_rreq
dnc_dwu_radr
diu_dnc_rack
diu_dnc_rvalid
diu_data
17
64
HCU
hcu_dnc_data
6
hcu_dnc_avail
dnc_hcu_ready
ink replacement unit
iru_avail
iru_data
6
iru_dn_mask
6
edu_ready
error diffusion unit
36
17
17
6
plane[5-0]_replace_pattern
dn_table_start_adr
dn_table_end_adr
diffuse_enable
max_dot
16
edu_dn_mask
edu_data
6
6
configuration registers
fixative_mask1
6
fixative_mask2
6
fixative correction unit
7
32
32
Dead Nozzle Compensator
6
pcu_dnc_sel
pcu_rwn
pcu_adr
pcu_dataout
dnc_pcu_rdy
dnc_pcu_datain
dnc_dwu_data
dnc_dwu_avail
dwu_dnc_ready
PEP Controller Unit
Dotline Writer Unit

FIG. 247

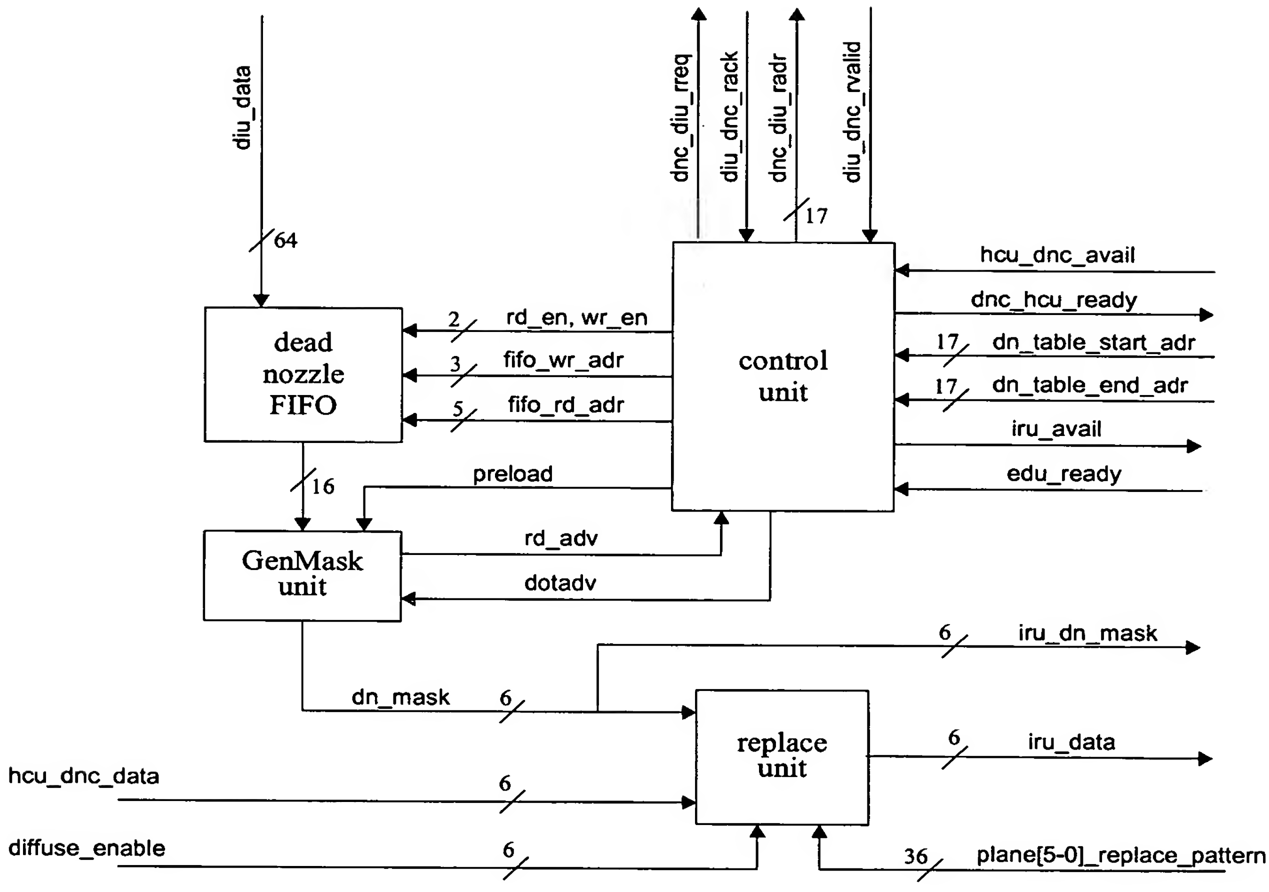
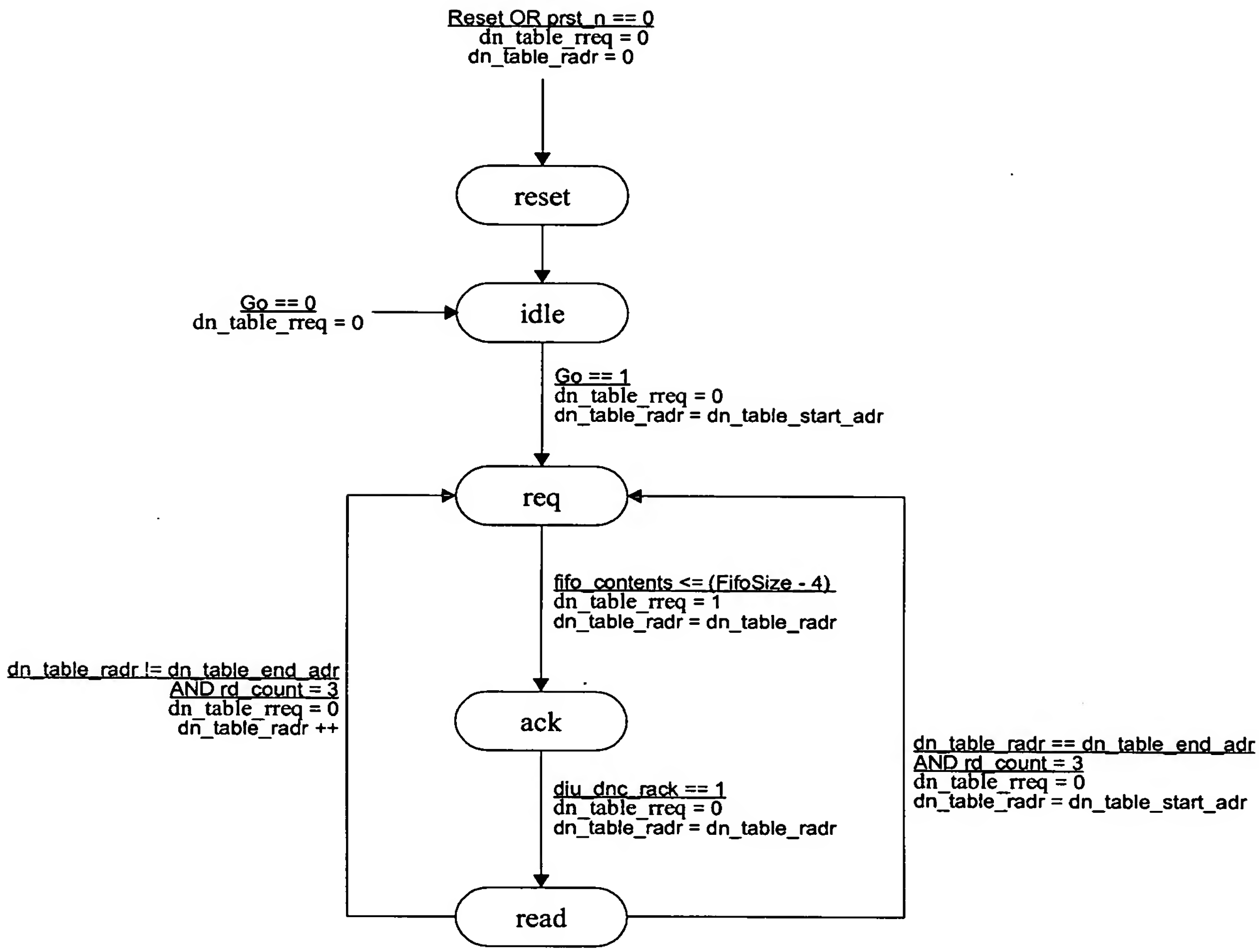

diu_data
64
dnc_diu_rreq
diu_dnc_rack
dnc_diu_radr
diu_dnc_rvalid
17
hcu_dnc_avail
dnc_hcu_ready
17
dn_table_start_adr
17
dn_table_end_adr
iru_avail
edu_ready
dead nozzle FIFO
2
rd_en, wr_en
3
fifo_wr_adr
5
fifo_rd_adr
control unit
16
preload
GenMask unit
rd_adv
dotadv
6
iru_dn_mask
dn_mask
6
replace unit
6
iru_data
hcu_dnc_data
6
diffuse_enable
6
36
plane[5-0]_replace_pattern

FIG. 248



Reset OR prst_n == 0
dn_table_rreq = 0
dn_table_radr = 0
reset
Go == 0
dn_table_rreq = 0
idle
Go == 1
dn_table_rreq = 0
dn_table_radr = dn_table_start_adr
req
fifo_contents <= (FifoSize - 4)
dn_table_rreq = 1
dn_table_radr = dn_table_radr
dn_table_radr != dn_table_end_adr
AND rd_count = 3
dn_table_rreq = 0
dn_table_radr ++
ack
dn_table_radr == dn_table_end_adr
AND rd_count = 3
dn_table_rreq = 0
dn_table_radr = dn_table_start_adr
diu_dnc_rack == 1
dn_table_rreq = 0
dn_table_radr = dn_table_radr
read

FIG. 249

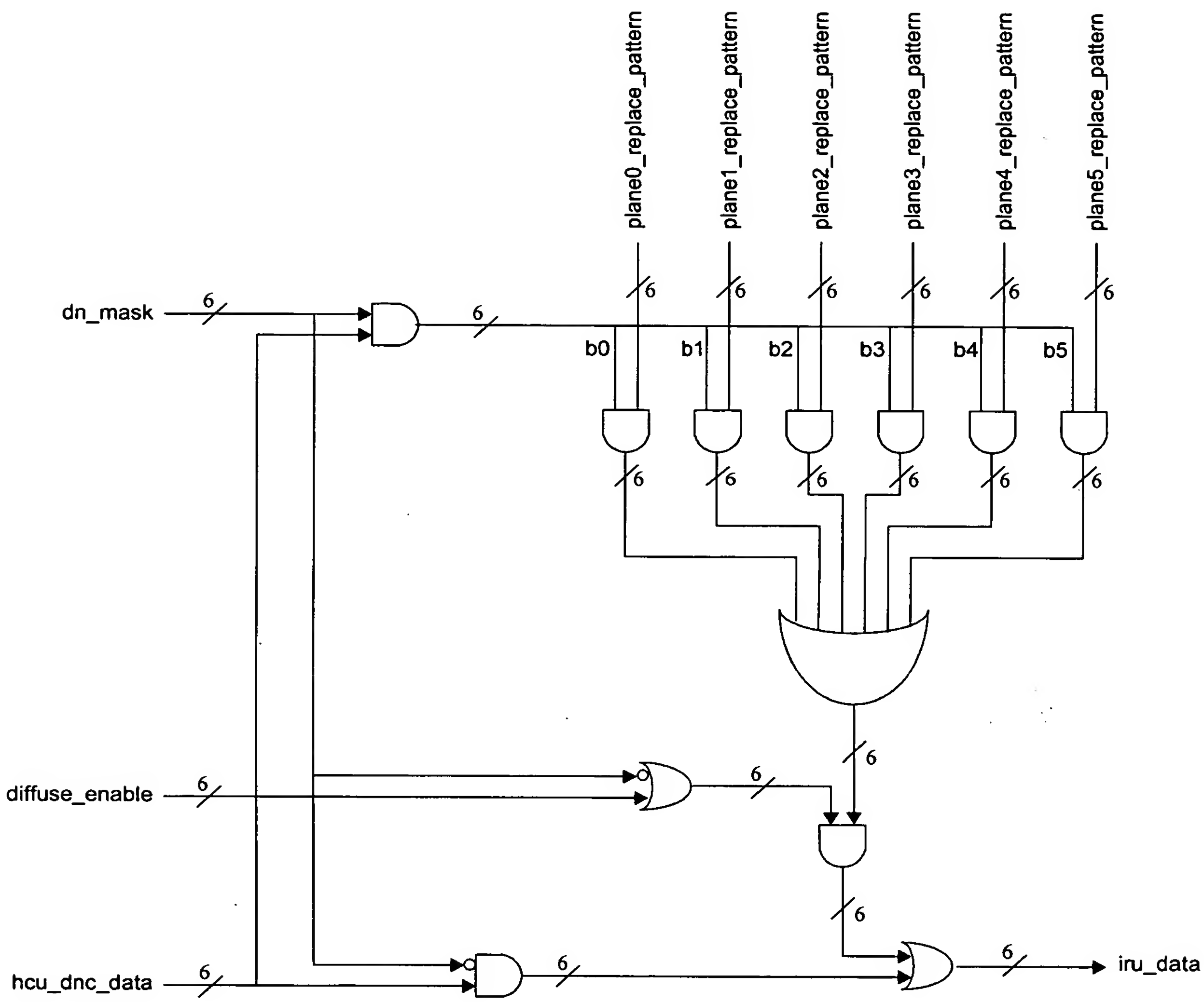
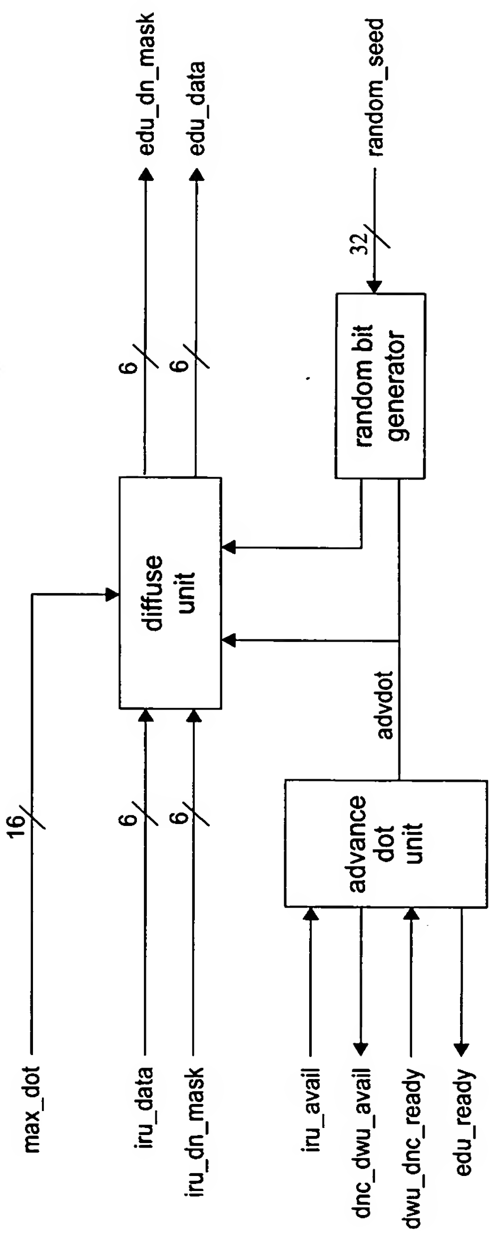

plane0_replace_pattern
plane1_replace_pattern
plane2_replace_pattern
plane3_replace_pattern
plane4_replace_pattern
plane5_replace_pattern
dn_mask
6
b0
b1
b2
b3
b4
b5
diffuse_enable
hcu_dnc_data
iru_data

FIG. 250



max_dot
16
iru_data
6
iru_dn_mask
6
diffuse unit
edu_dn_mask
6
edu_data
6
random bit generator
random_seed
32
advdot
advance dot unit
iru_avail
dnc_dwu_avail
dwu_dnc_ready
edu_ready

FIG. 251

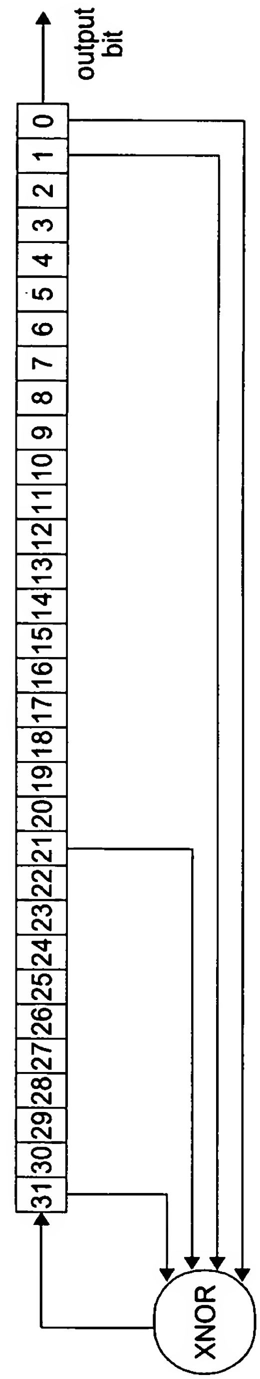


FIG. 252

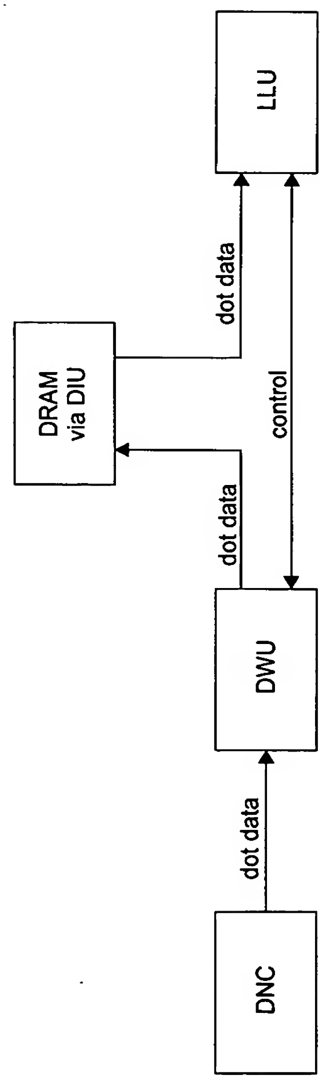
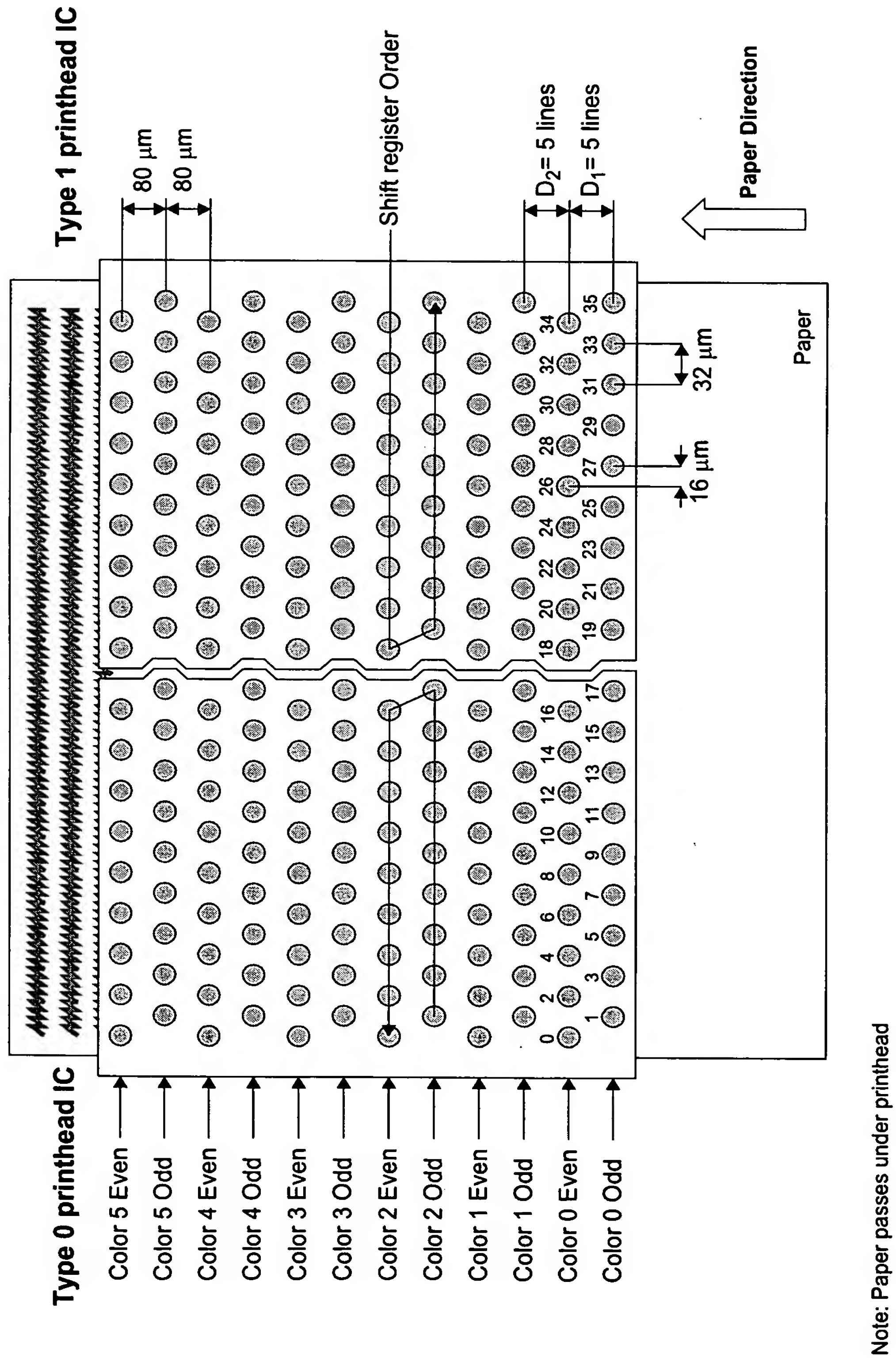
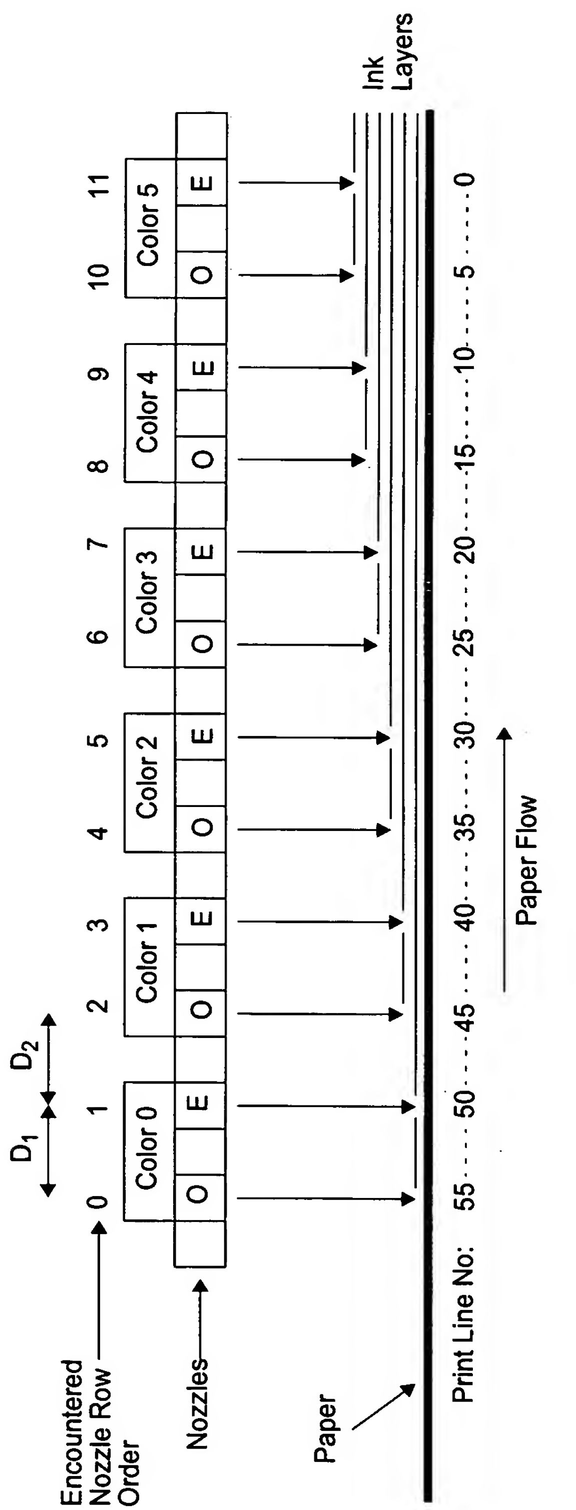


FIG. 253



Type 1 printhead IC
Type 0 printhead IC
80 µm
80 µm
Shift register Order
D2= 5 lines
D1= 5 lines
Paper Direction
Paper
32 µm
16 µm
0 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17
18 19 20 21 22 23 24 25 26 27 28 29 30 31 32 33 34 35
Color 5 Even
Color 5 Odd
Color 4 Even
Color 4 Odd
Color 3 Even
Color 3 Odd
Color 2 Even
Color 2 Odd
Color 1 Even
Color 1 Odd
Color 0 Even
Color 0 Odd
Note: Paper passes under printhead

FIG. 254



Encountered Nozzle Row Order
0
1
2
3
4
5
6
7
8
9
10
11
D_1
D_2
Color 0
Color 1
Color 2
Color 3
Color 4
Color 5
O
E
Nozzles
Ink Layers
Paper
Print Line No:
55
50
45
40
35
30
25
20
15
10
5
0
Paper Flow

FIG. 255

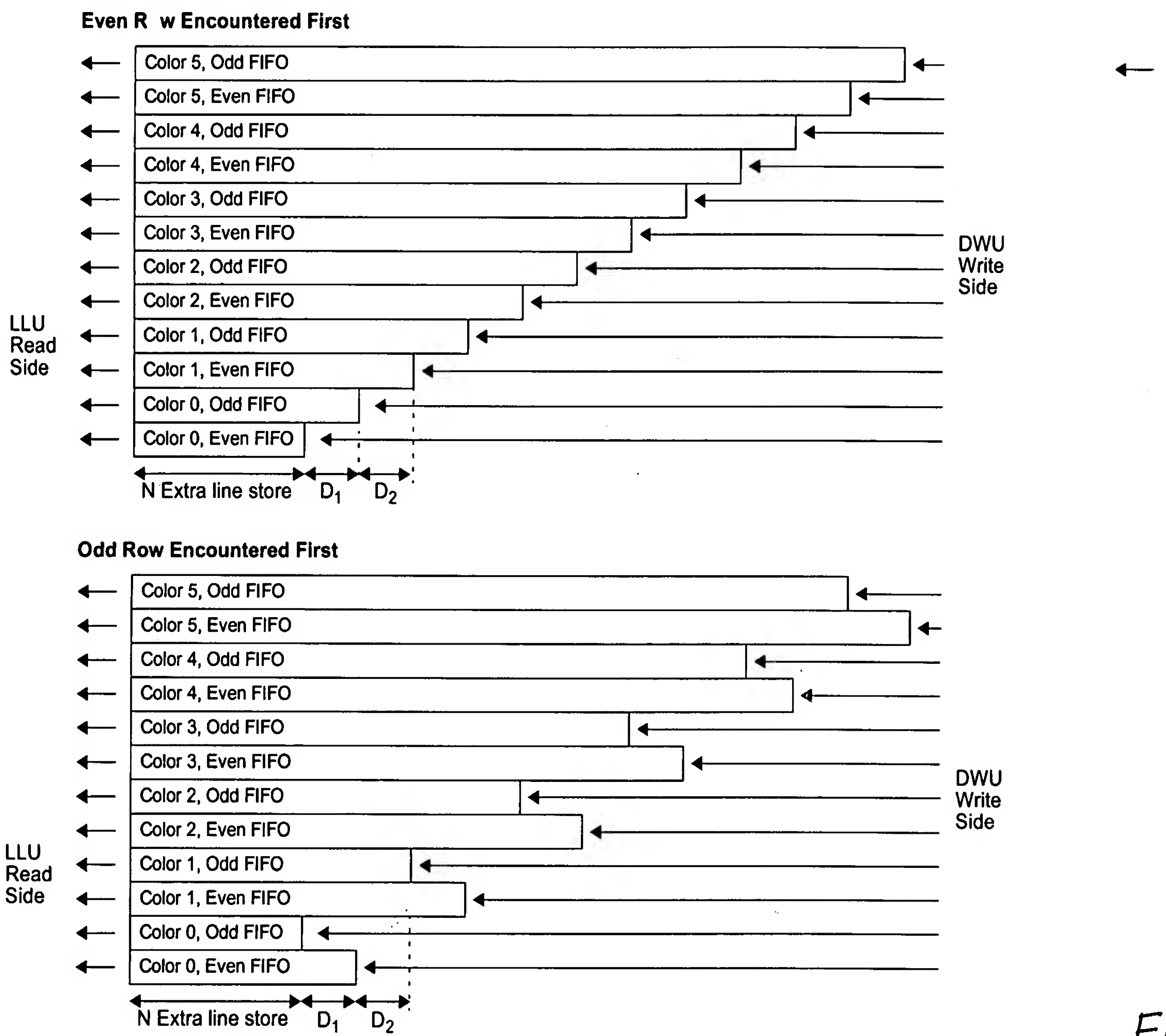

Even R w Encountered First
Color 5, Odd FIFO
Color 5, Even FIFO
Color 4, Odd FIFO
Color 4, Even FIFO
Color 3, Odd FIFO
Color 3, Even FIFO
Color 2, Odd FIFO
Color 2, Even FIFO
Color 1, Odd FIFO
Color 1, Even FIFO
Color 0, Odd FIFO
Color 0, Even FIFO
LLU Read Side
DWU Write Side
N Extra line store
D1
D2
Odd Row Encountered First
Color 5, Odd FIFO
Color 5, Even FIFO
Color 4, Odd FIFO
Color 4, Even FIFO
Color 3, Odd FIFO
Color 3, Even FIFO
Color 2, Odd FIFO
Color 2, Even FIFO
Color 1, Odd FIFO
Color 1, Even FIFO
Color 0, Odd FIFO
Color 0, Even FIFO
LLU Read Side
DWU Write Side
N Extra line store
D1
D2

FIG. 256

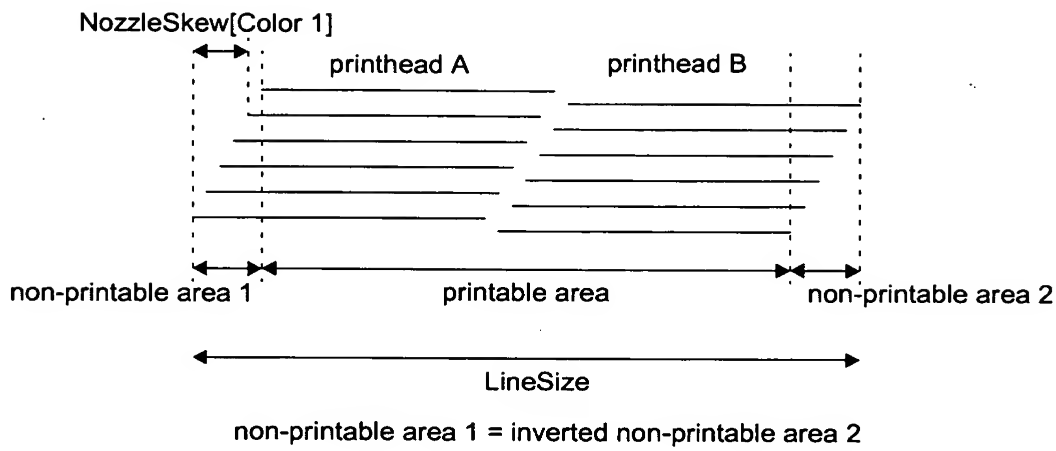


FIG. 257

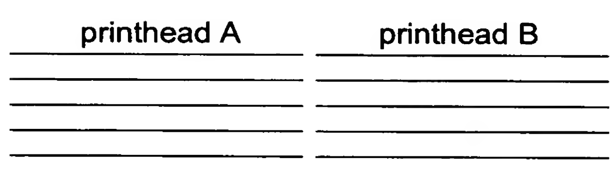
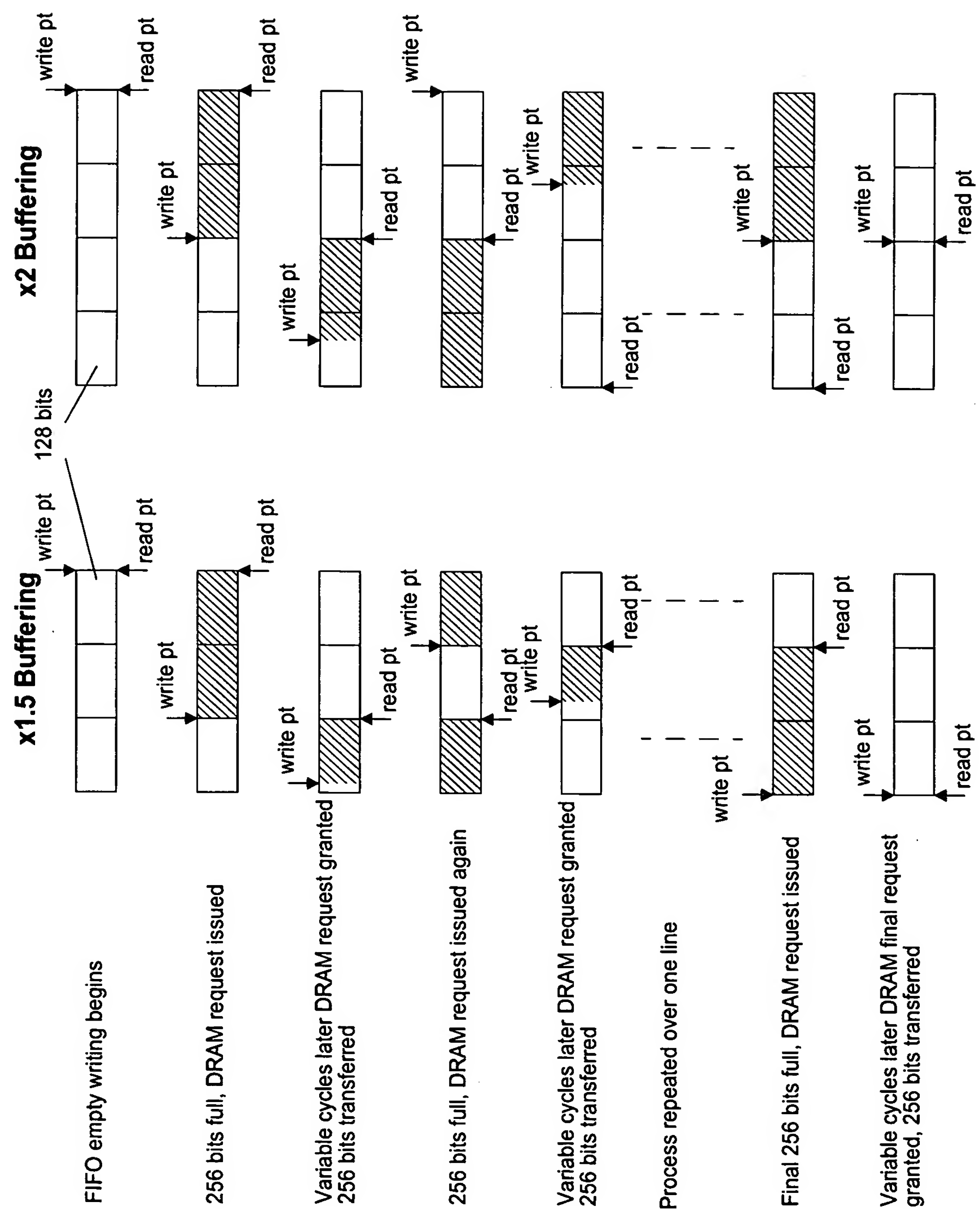


FIG. 258



x1.5 Buffering
x2 Buffering
128 bits
write pt
read pt
FIFO empty writing begins
256 bits full, DRAM request issued
Variable cycles later DRAM request granted 256 bits transferred
256 bits full, DRAM request issued again
Variable cycles later DRAM request granted 256 bits transferred
Process repeated over one line
Final 256 bits full, DRAM request issued
Variable cycles later DRAM final request granted, 256 bits transferred

FIG. 259

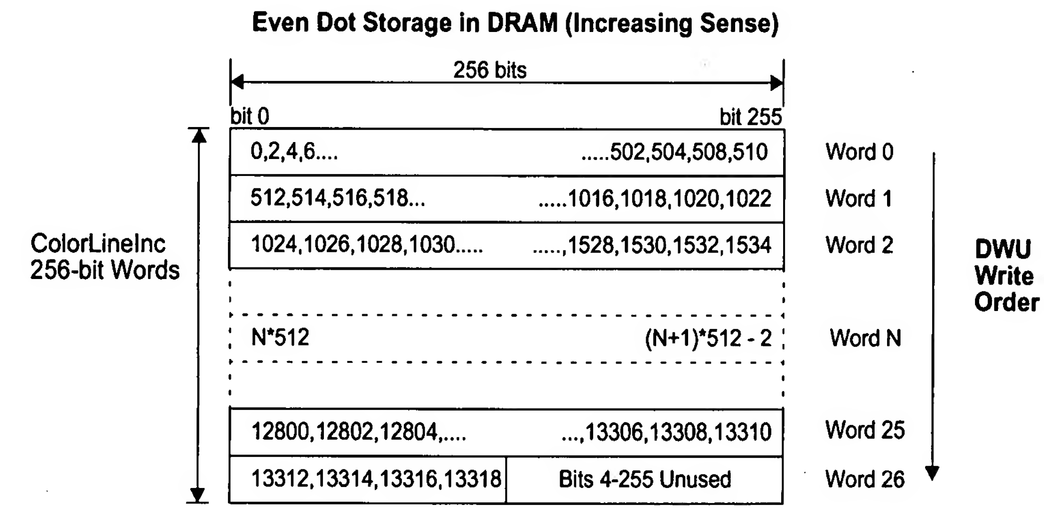


FIG. 260

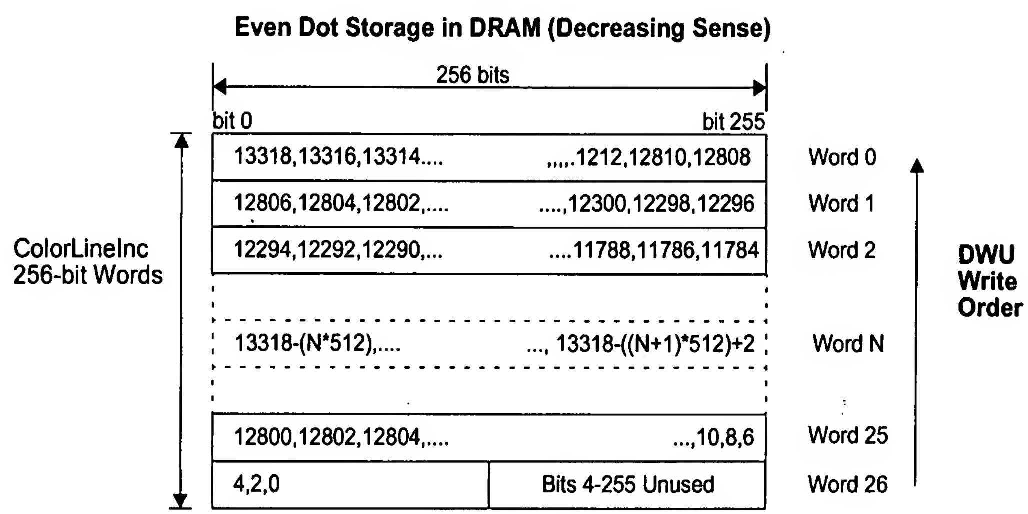


FIG. 261

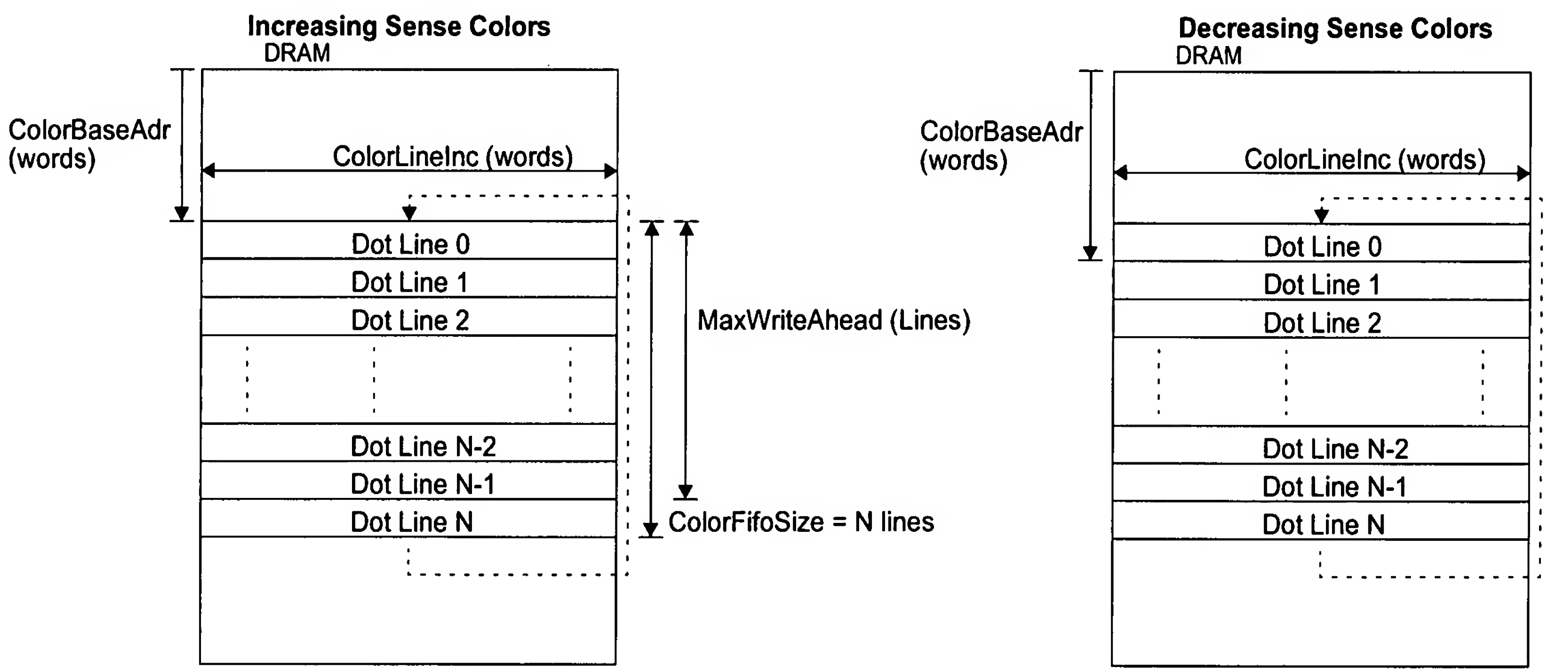
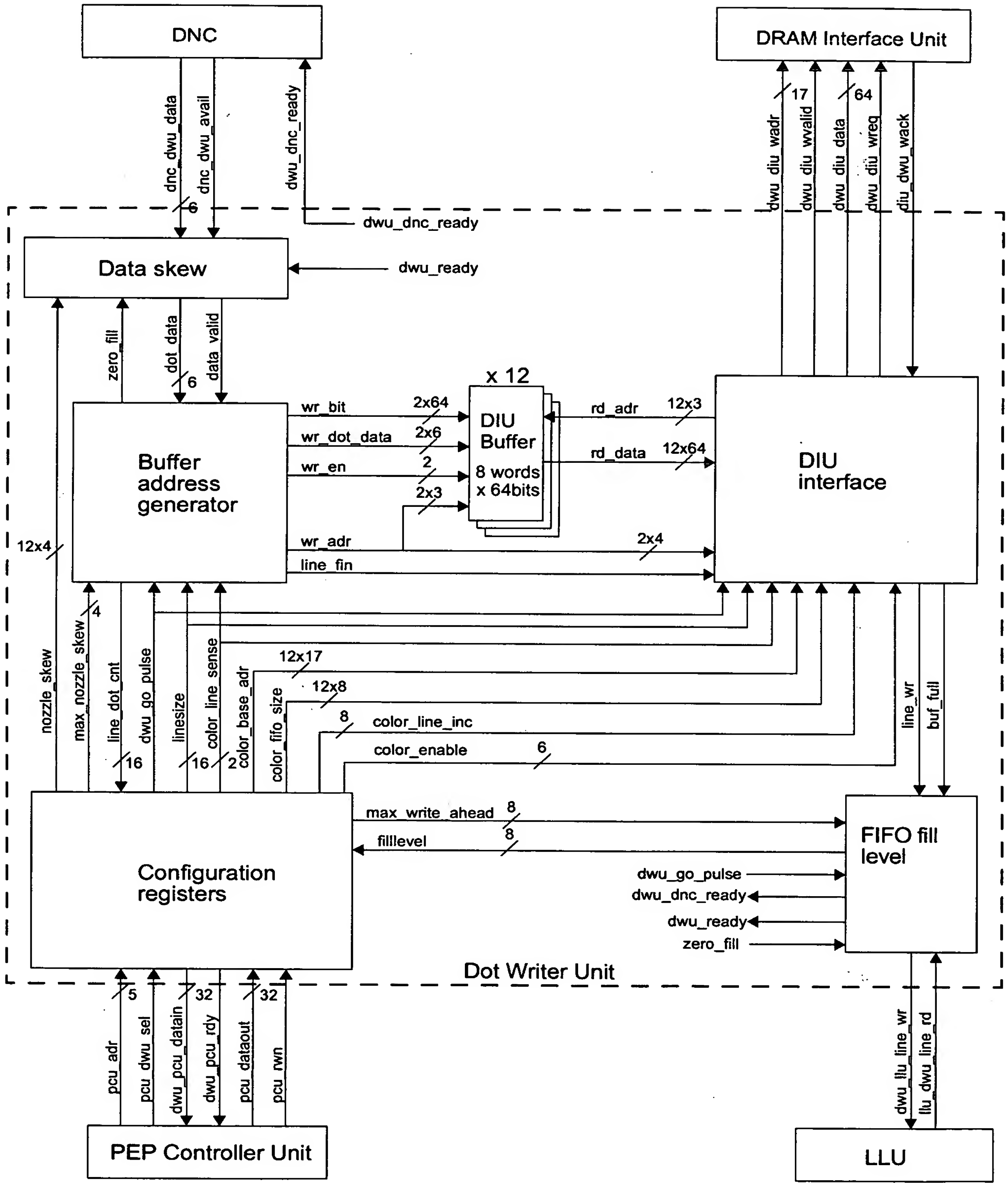

Increasing Sense Colors
DRAM
ColorBaseAdr (words)
ColorLineInc (words)
Dot Line 0
Dot Line 1
Dot Line 2
Dot Line N-2
Dot Line N-1
Dot Line N
MaxWriteAhead (Lines)
ColorFifoSize = N lines
Decreasing Sense Colors
DRAM
ColorBaseAdr (words)
ColorLineInc (words)
Dot Line 0
Dot Line 1
Dot Line 2
Dot Line N-2
Dot Line N-1
Dot Line N

FIG. 262



DNC
DRAM Interface Unit
dnc_dwu_data
dnc_dwu_avail
dwu_dnc_ready
dwu_diu_wadr
dwu_diu_wvalid
dwu_diu_data
dwu_diu_wreq
diu_dwu_wack
17
64
6
dwu_dnc_ready
Data skew
dwu_ready
zero_fill
dot_data
data_valid
6
x 12
DIU Buffer
8 words x 64bits
wr_bit
2x64
wr_dot_data
2x6
wr_en
2
2x3
rd_adr
12x3
rd_data
12x64
Buffer address generator
DIU interface
wr_adr
2x4
line_fin
12x4
4
nozzle_skew
max_nozzle_skew
line_dot_cnt
dwu_go_pulse
linesize
color_line_sense
color_base_adr
color_fifo_size
12x17
12x8
8
color_line_inc
color_enable
6
16
16
2
line_wr
buf_full
Configuration registers
max_write_ahead
8
filllevel
8
FIFO fill level
dwu_go_pulse
dwu_dnc_ready
dwu_ready
zero_fill
Dot Writer Unit
5
32
32
pcu_adr
pcu_dwu_sel
dwu_pcu_datain
dwu_pcu_rdy
pcu_dataout
pcu_rwn
dwu_llu_line_wr
llu_dwu_line_rd
PEP Controller Unit
LLU

FIG. 263

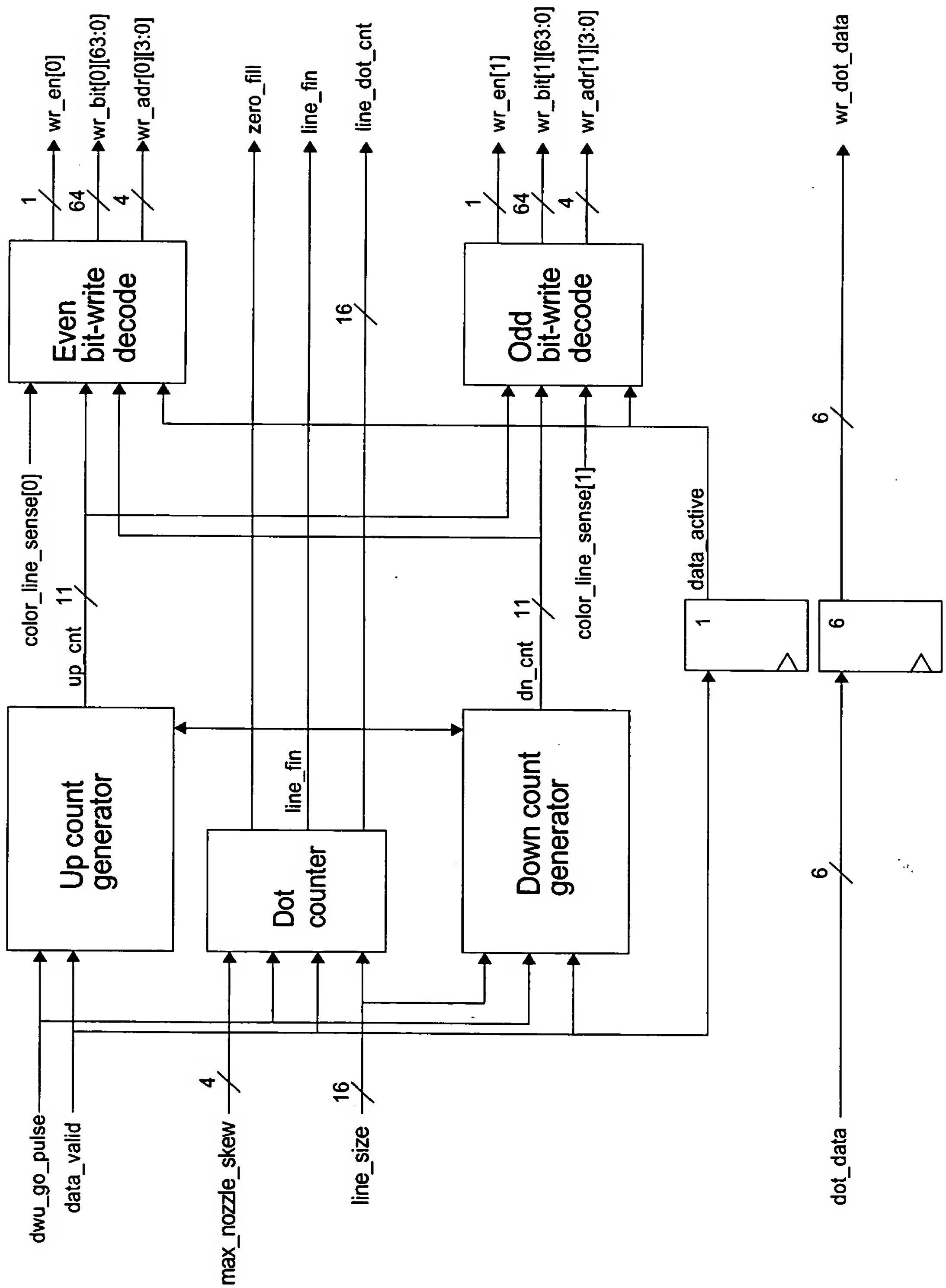

Up count generator
Dot counter
Down count generator
Even bit-write decode
Odd bit-write decode
dwu_go_pulse
data_valid
max_nozzle_skew
line_size
dot_data
up_cnt
dn_cnt
line_fin
color_line_sense[0]
color_line_sense[1]
data_active
wr_en[0]
wr_bit[0][63:0]
wr_adr[0][3:0]
zero_fill
line_fin
line_dot_cnt
wr_en[1]
wr_bit[1][63:0]
wr_adr[1][3:0]
wr_dot_data
1
64
4
16
11
6

FIG. 264

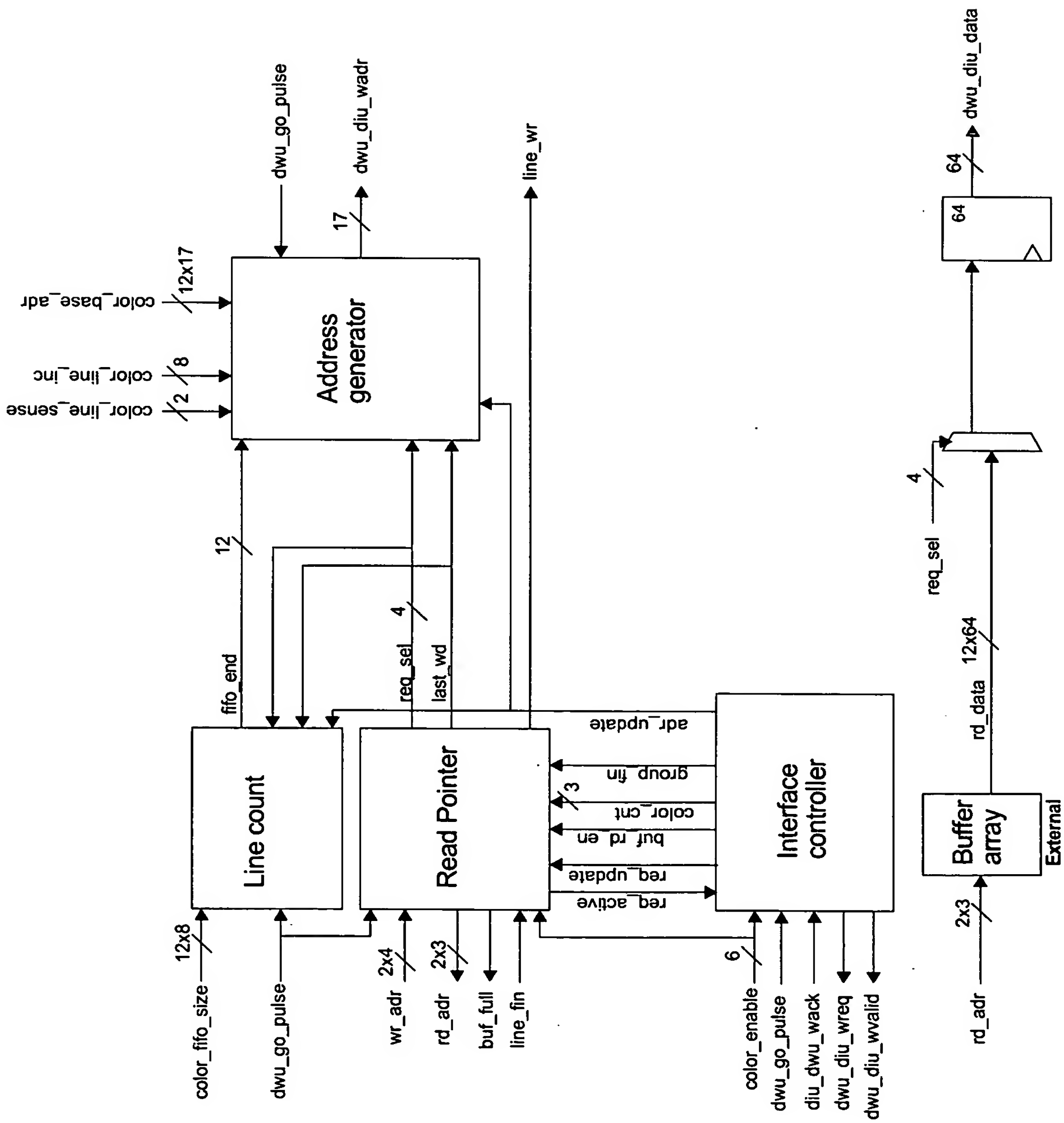
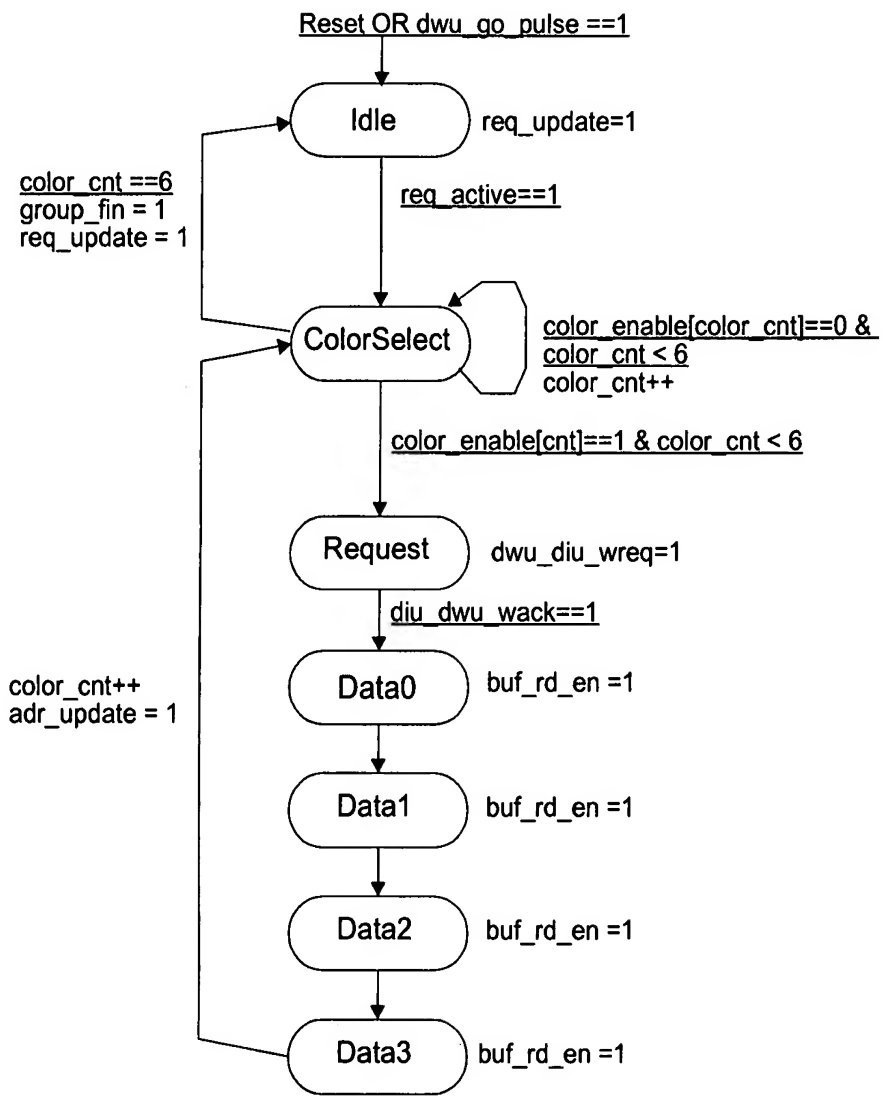

Address generator
Line count
Read Pointer
Interface controller
Buffer array
External
color_base_adr
color_line_inc
color_line_sense
12x17
8
2
dwu_go_pulse
dwu_diu_wadr
17
line_wr
12
fifo_end
4
req_sel
last_wd
adr_update
group_fin
color_cnt
3
buf_rd_en
req_update
req_active
12x8
color_fifo_size
dwu_go_pulse
2x4
wr_adr
2x3
rd_adr
buf_full
line_fin
6
color_enable
dwu_go_pulse
diu_dwu_wack
dwu_diu_wreq
dwu_diu_wvalid
2x3
rd_adr
rd_data
12x64
req_sel
4
64
64
dwu_diu_data

FIG. 265



Machine remains in same state by default
All outputs are zero unless otherwise stated

State Description:

Idle:	Idle state wait for active request
ColorSelect:	Select the color to update before requesting to DIU
Request:	Request issued wait for acknowledge
Data0:	Data word 0 transfer
Data1:	Data word 1 transfer
Data2:	Data word 2 transfer
Data3:	Data word 3 transfer

FIG. 266

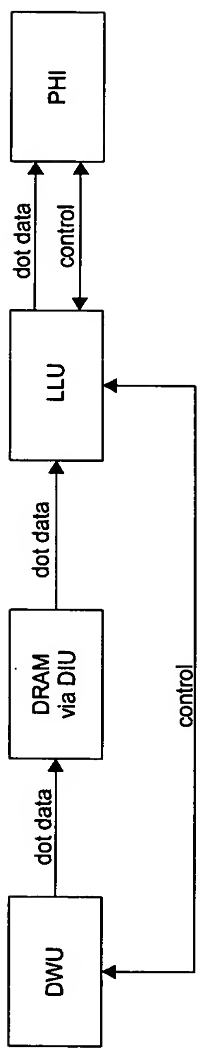


FIG. 267

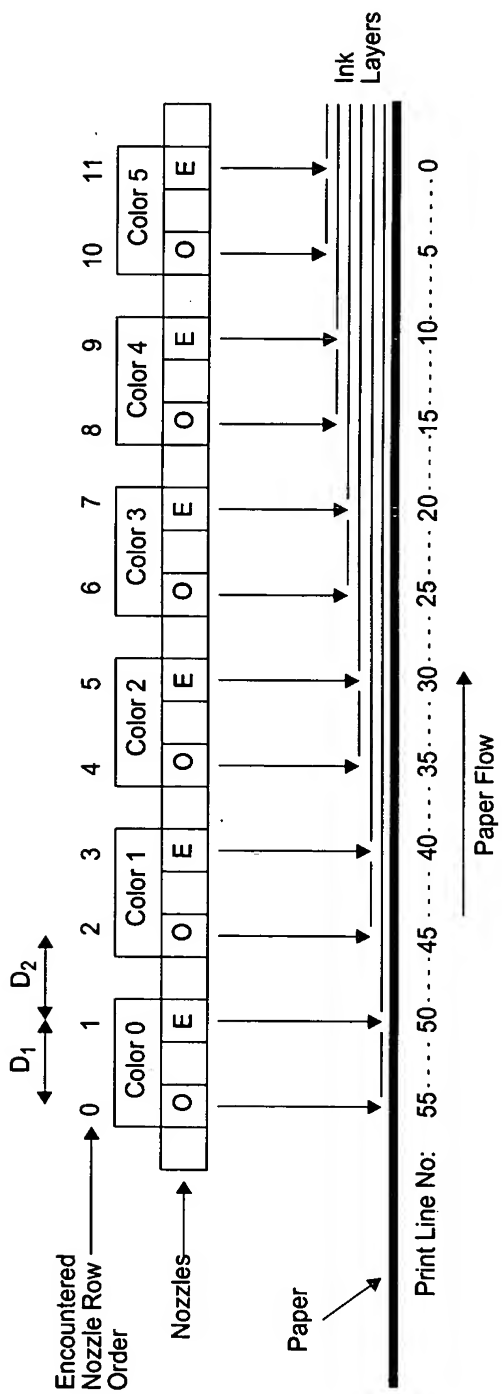


FIG. 268

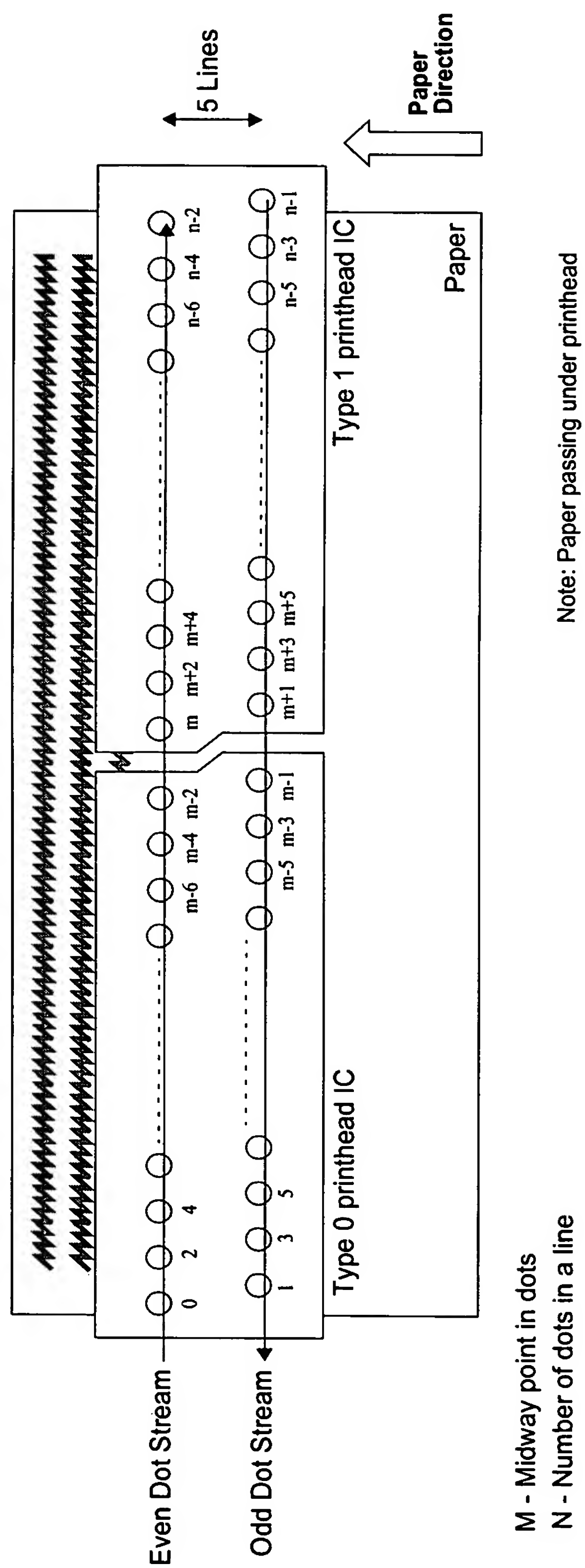
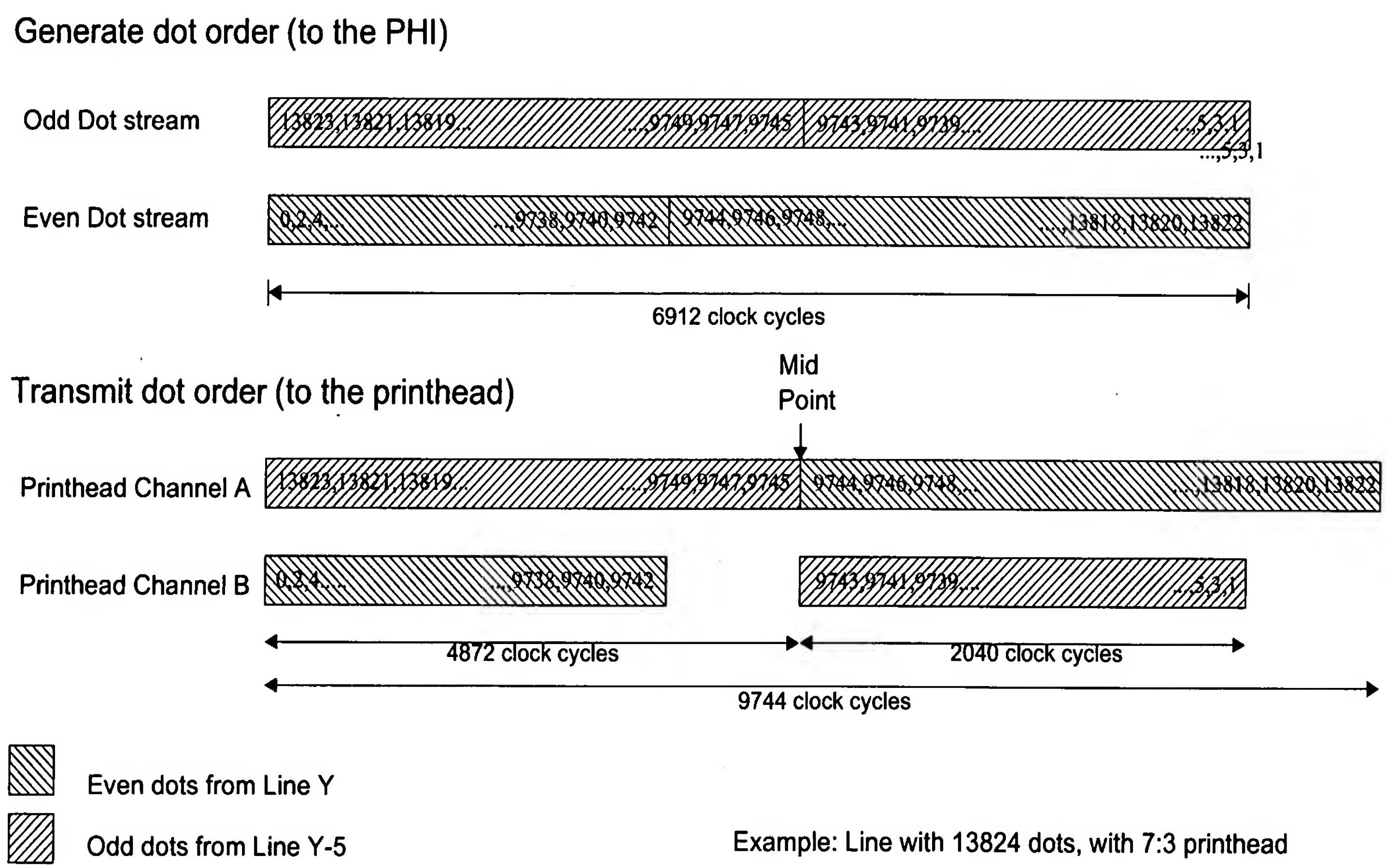

5 Lines
Paper Direction
Paper
Type 1 printhead IC
Type 0 printhead IC
Even Dot Stream
Odd Dot Stream
0 2 4
1 3 5
m-6 m-4 m-2
m-5 m-3 m-1
m m+2 m+4
m+1 m+3 m+5
n-6 n-4 n-2
n-5 n-3 n-1
M - Midway point in dots
N - Number of dots in a line
Note: Paper passing under printhead

FIG. 269



Generate dot order (to the PHI)
Odd Dot stream
13823,13821,13819...
...,9749,9747,9745
9743,9741,9739,...
...,5,3,1
Even Dot stream
0,2,4,...
...,9738,9740,9742
9744,9746,9748,...
...,13818,13820,13822
6912 clock cycles
Transmit dot order (to the printhead)
Mid Point
Printhead Channel A
13823,13821,13819...
...,9749,9747,9745
9744,9746,9748,...
...,13818,13820,13822
Printhead Channel B
0,2,4,...
...,9738,9740,9742
9743,9741,9739,...
...,5,3,1
4872 clock cycles
2040 clock cycles
9744 clock cycles
Even dots from Line Y
Odd dots from Line Y-5
Example: Line with 13824 dots, with 7:3 printhead

FIG. 270

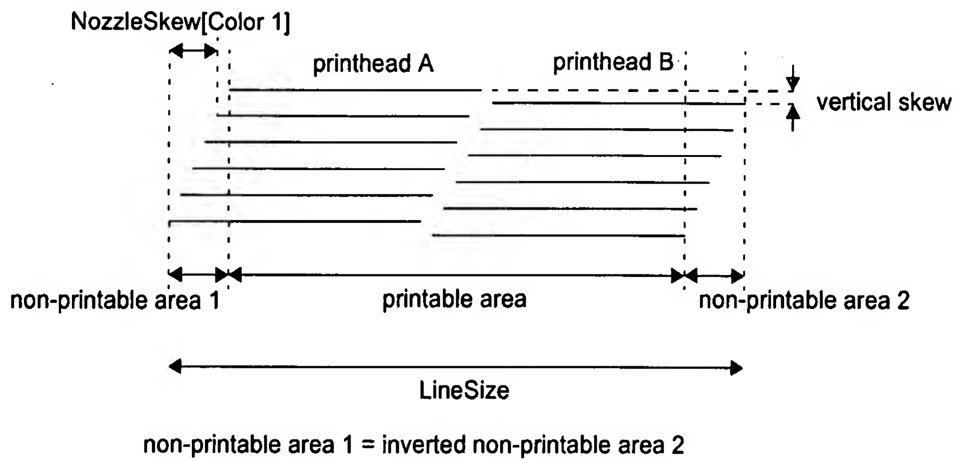

NozzleSkew[Color 1]
printhead A
printhead B
vertical skew
non-printable area 1
printable area
non-printable area 2
LineSize
non-printable area 1 = inverted non-printable area 2

FIG. 271

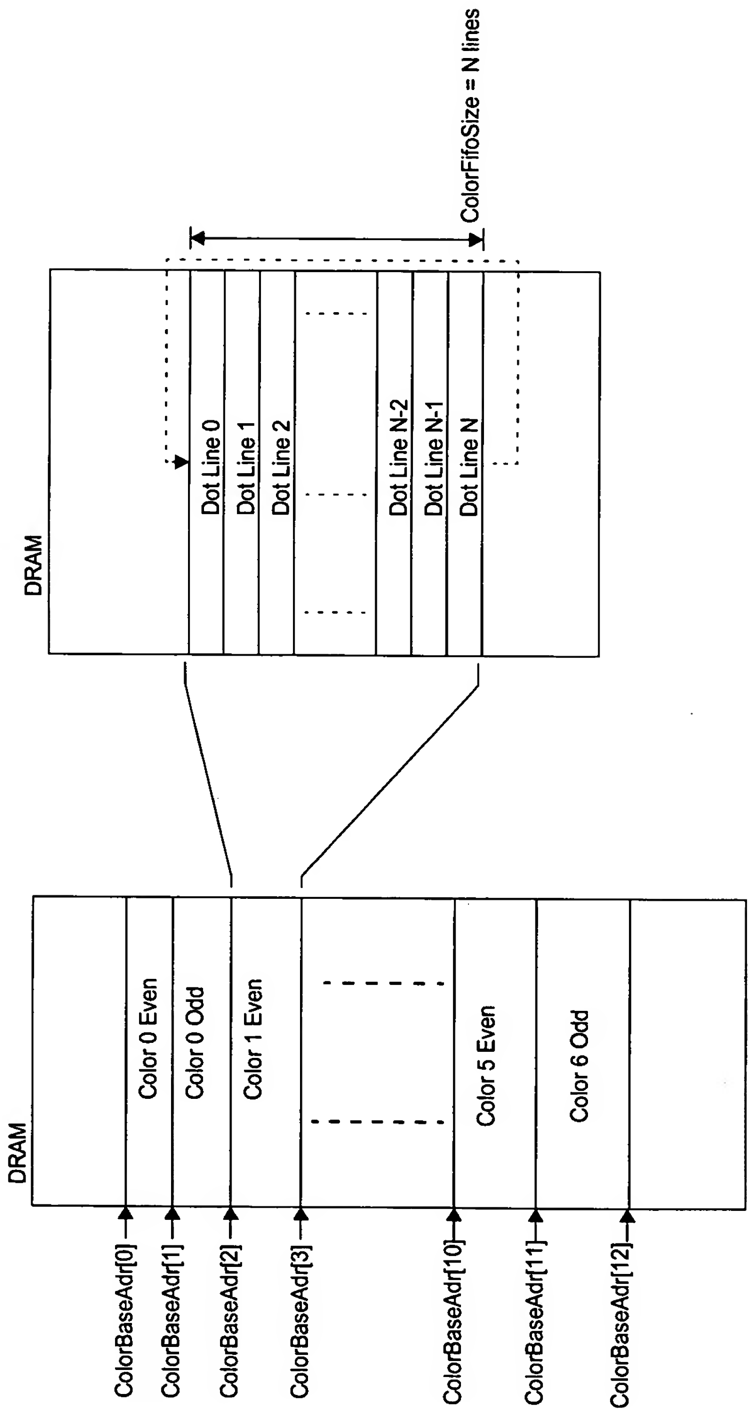

ColorFifoSize = N lines
DRAM
Dot Line 0
Dot Line 1
Dot Line 2
Dot Line N-2
Dot Line N-1
Dot Line N
DRAM
Color 0 Even
Color 0 Odd
Color 1 Even
Color 5 Even
Color 6 Odd
ColorBaseAdr[0]
ColorBaseAdr[1]
ColorBaseAdr[2]
ColorBaseAdr[3]
ColorBaseAdr[10]
ColorBaseAdr[11]
ColorBaseAdr[12]

FIG. 272

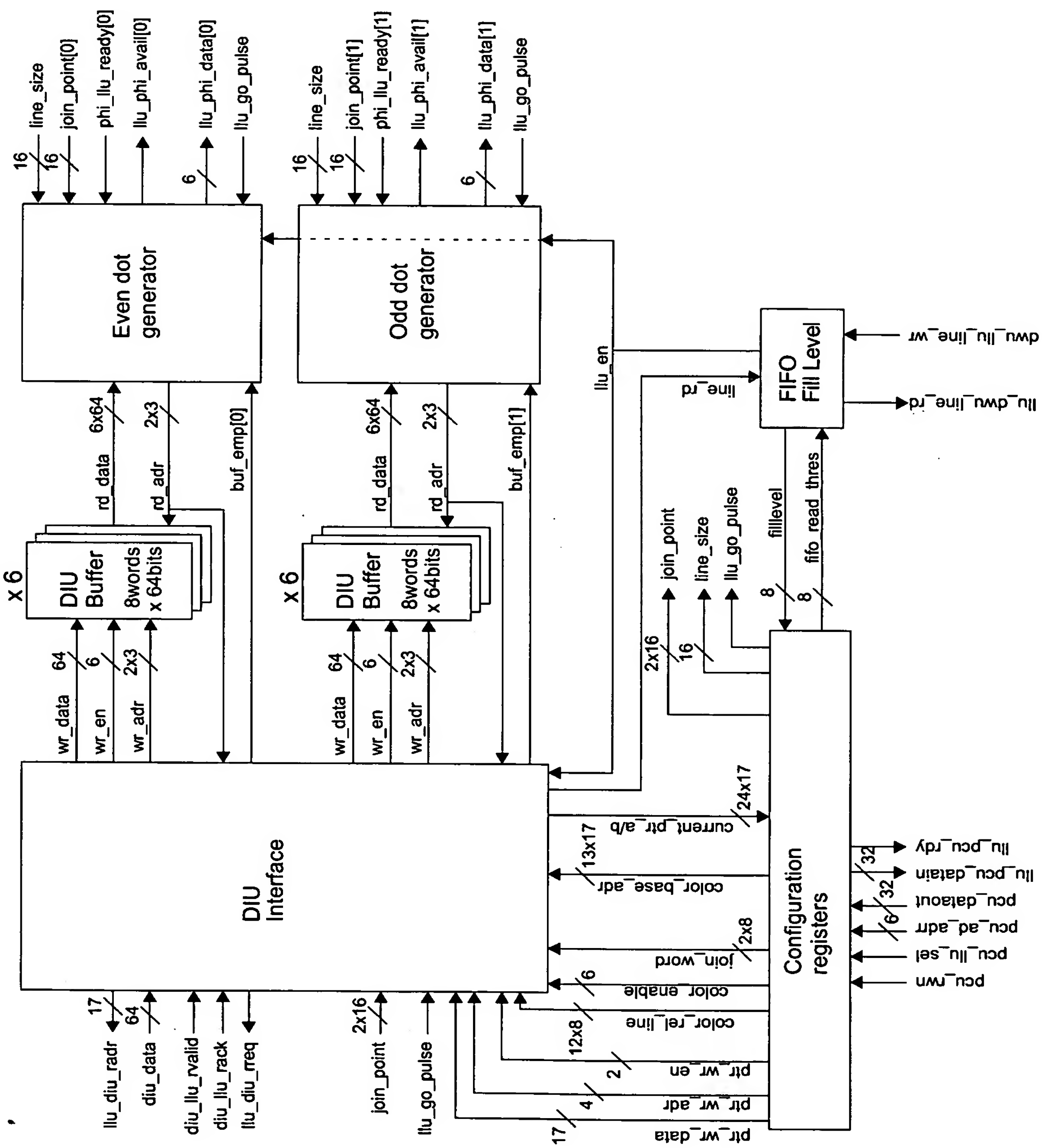
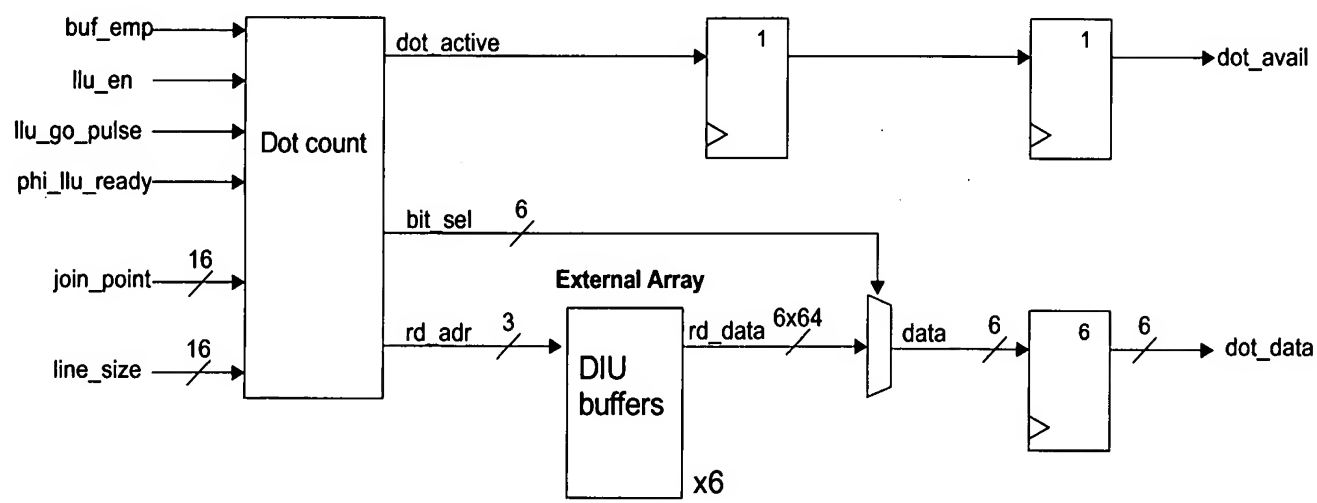
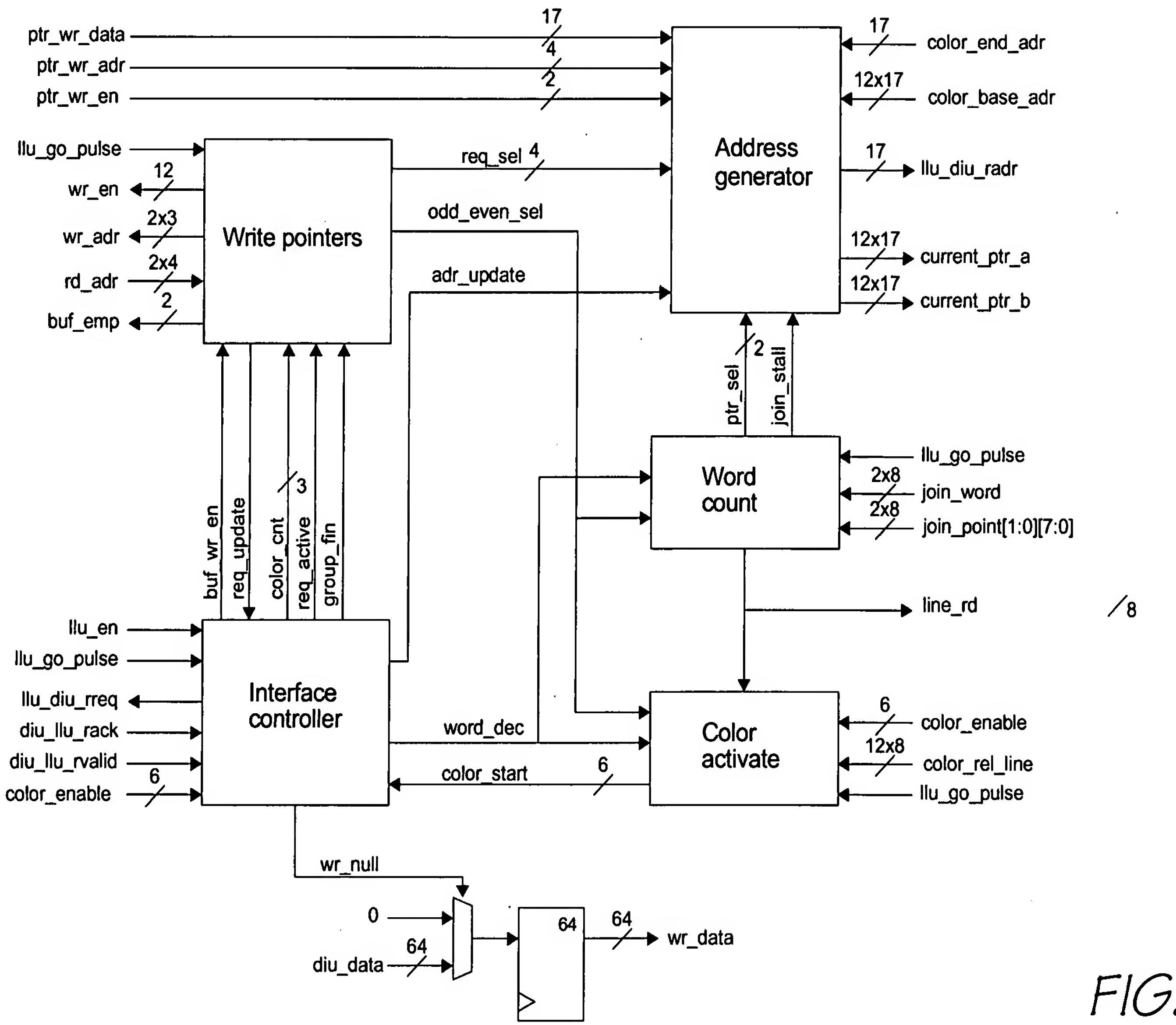

Even dot generator
Odd dot generator
DIU Buffer 8words x 64bits
x 6
DIU Interface
FIFO Fill Level
Configuration registers
line_size
join_point[0]
phi_llu_ready[0]
llu_phi_avail[0]
llu_phi_data[0]
llu_go_pulse
join_point[1]
phi_llu_ready[1]
llu_phi_avail[1]
llu_phi_data[1]
rd_data
rd_adr
buf_emp[0]
buf_emp[1]
wr_data
wr_en
wr_adr
llu_en
line_rd
dwu_llu_line_wr
llu_dwu_line_rd
filllevel
fifo_read_thres
join_point
llu_go_pulse
current_ptr_a/b
color_base_adr
join_word
color_enable
color_rel_line
ptr_wr_en
ptr_wr_adr
ptr_wr_data
llu_pcu_rdy
llu_pcu_datain
pcu_dataout
pcu_pe_adr
pcu_llu_sel
pcu_rwn
llu_diu_radr
diu_data
diu_llu_rvalid
diu_llu_rack
llu_diu_rreq
16
6
6x64
2x3
64
8
2x16
24x17
13x17
2x8
12x8
2
4
17
32

FIG. 273



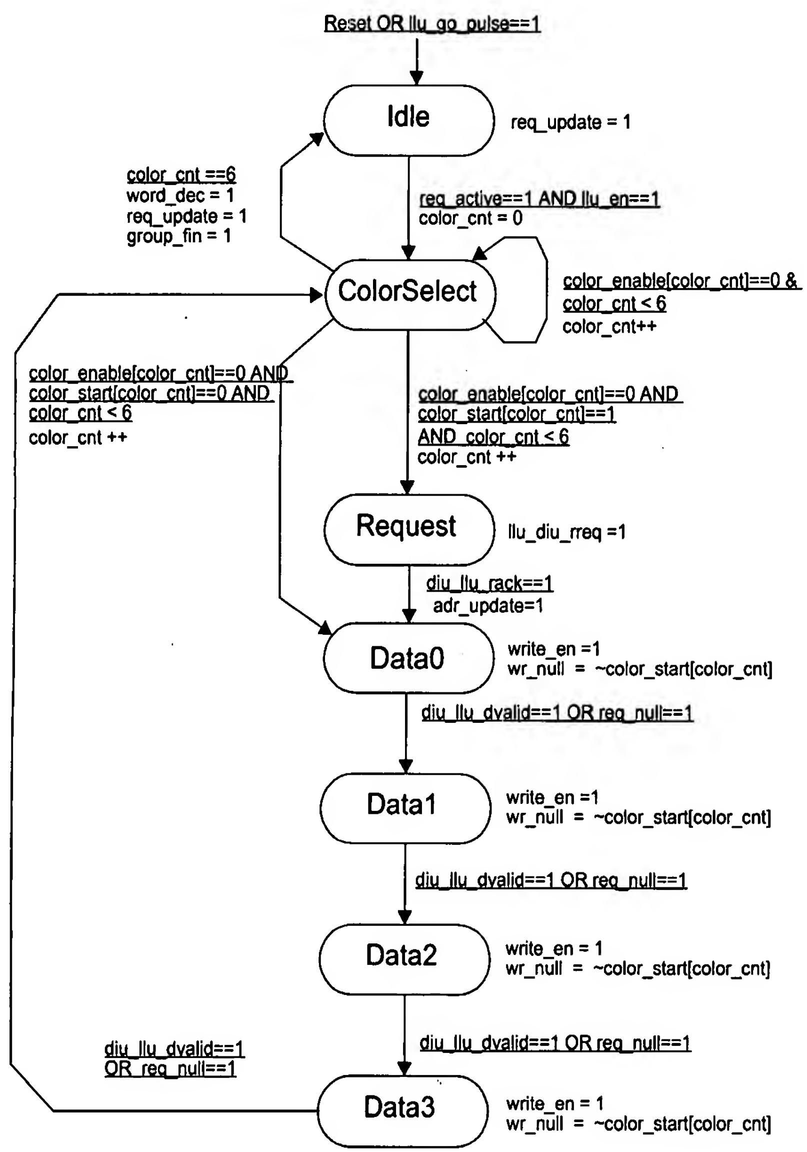
buf_emp
llu_en
llu_go_pulse
phi_llu_ready
join_point
16
line_size
16
Dot count
dot_active
bit_sel
6
rd_adr
3
External Array
DIU
buffers
x6
rd_data
6x64
data
6
1
1
dot_avail
6
6
dot_data

FIG. 274



ptr_wr_data
ptr_wr_adr
ptr_wr_en
17
4
2
Address generator
17
color_end_adr
12x17
color_base_adr
17
llu_diu_radr
12x17
current_ptr_a
12x17
current_ptr_b
llu_go_pulse
12
wr_en
2x3
wr_adr
2x4
rd_adr
2
buf_emp
Write pointers
req_sel 4
odd_even_sel
adr_update
ptr_sel
2
join_stall
Word count
llu_go_pulse
2x8
join_word
2x8
join_point[1:0][7:0]
line_rd
8
buf_wr_en
req_update
color_cnt
3
req_active
group_fin
llu_en
llu_go_pulse
llu_diu_rreq
diu_llu_rack
diu_llu_rvalid
6
color_enable
Interface controller
word_dec
color_start
6
Color activate
6
color_enable
12x8
color_rel_line
llu_go_pulse
wr_null
0
64
diu_data
64
64
wr_data

FIG. 275



Machine remains in same state by default
All outputs are zero unless otherwise stated

State Description:

Idle:	Idle state wait for active request
ColorSelect:	Select the color to update before requesting to DIU
Request:	Request issued wait for acknowledge
Data0:	Data word 0 transfer
Data1:	Data word 1 transfer
Data2:	Data word 2 transfer
Data3:	Data word 3 transfer

FIG. 276

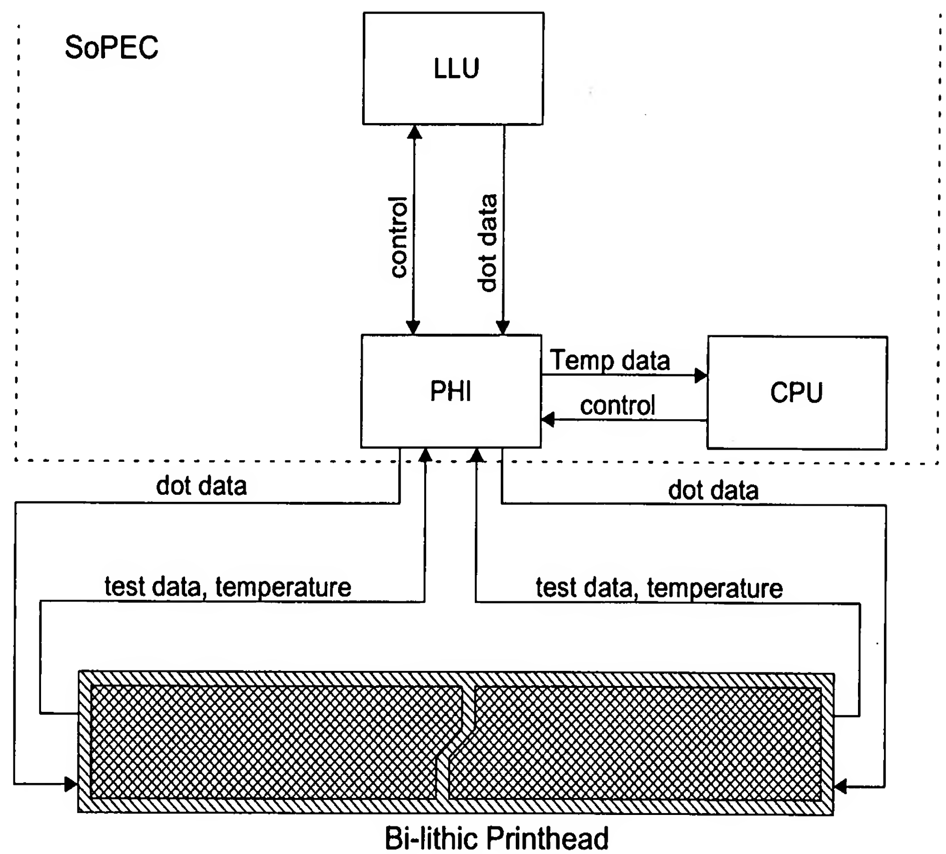

SoPEC
LLU
control
dot data
PHI
Temp data
control
CPU
dot data
dot data
test data, temperature
test data, temperature
Bi-lithic Printhead

FIG. 277

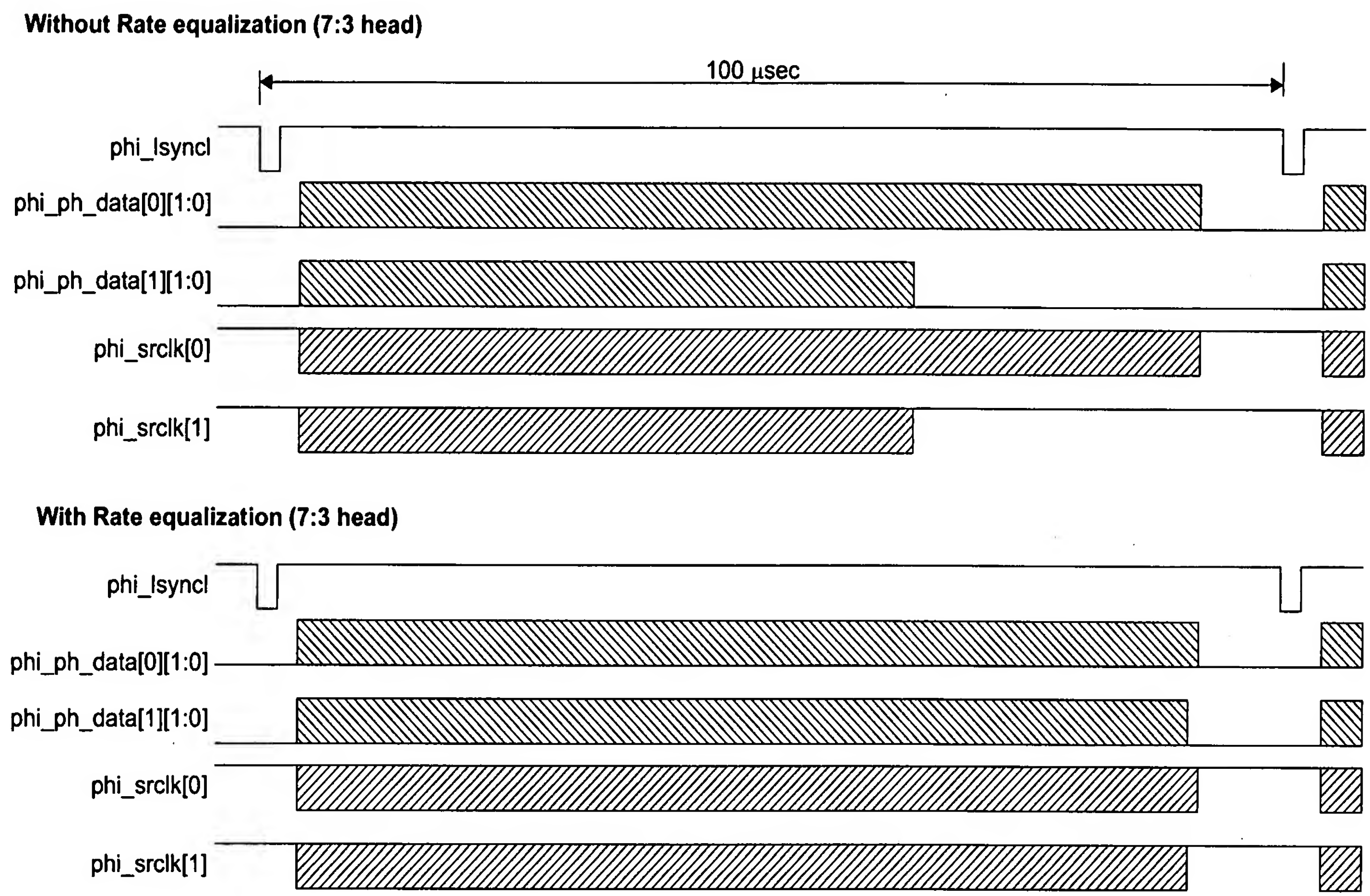
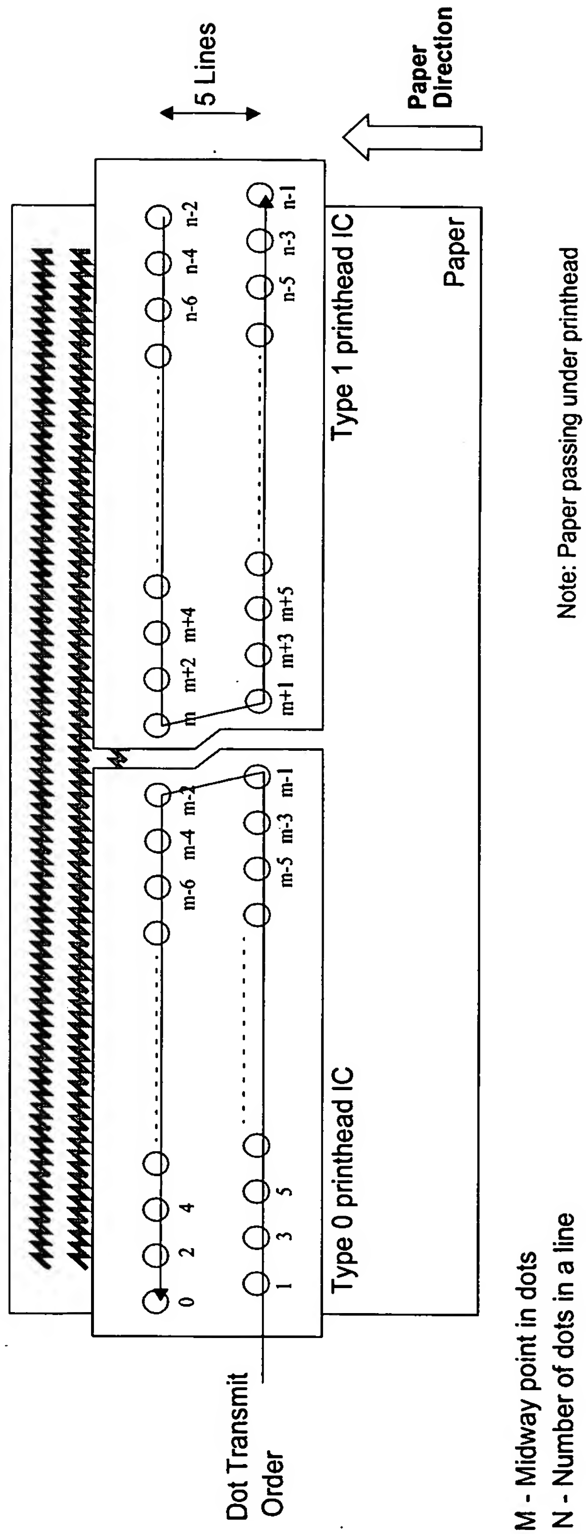

Without Rate equalization (7:3 head)
100 µsec
phi_lsyncl
phi_ph_data[0][1:0]
phi_ph_data[1][1:0]
phi_srclk[0]
phi_srclk[1]
With Rate equalization (7:3 head)
phi_lsyncl
phi_ph_data[0][1:0]
phi_ph_data[1][1:0]
phi_srclk[0]
phi_srclk[1]

FIG. 279



5 Lines
Paper Direction
Paper
Type 1 printhead IC
Type 0 printhead IC
0
2
4
1
3
5
m-6
m-4
m-2
m-5
m-3
m-1
m
m+2
m+4
m+1
m+3
m+5
n-6
n-4
n-2
n-5
n-3
n-1
Dot Transmit Order
M - Midway point in dots
N - Number of dots in a line
Note: Paper passing under printhead

FIG. 280

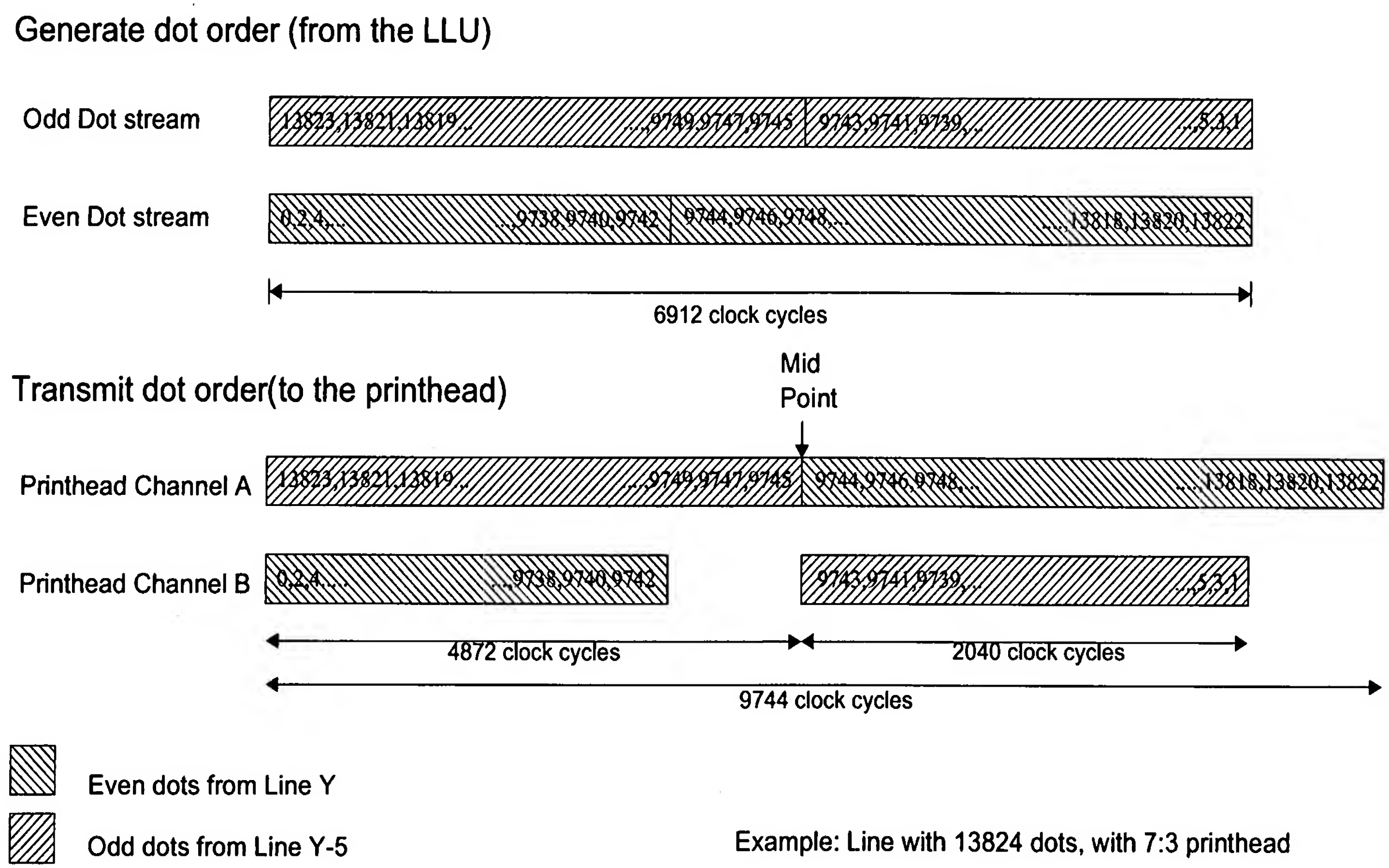

Generate dot order (from the LLU)
Odd Dot stream
13823,13821,13819...
...,9749,9747,9745
9743,9741,9739,...
...,5,3,1
Even Dot stream
0,2,4,...
...,9738,9740,9742
9744,9746,9748,...
...,13818,13820,13822
6912 clock cycles
Transmit dot order(to the printhead)
Mid Point
Printhead Channel A
13823,13821,13819...
...,9749,9747,9745
9744,9746,9748,...
...,13818,13820,13822
Printhead Channel B
0,2,4,...
...,9738,9740,9742
9743,9741,9739,...
...,5,3,1
4872 clock cycles
2040 clock cycles
9744 clock cycles
Even dots from Line Y
Odd dots from Line Y-5
Example: Line with 13824 dots, with 7:3 printhead

FIG. 281

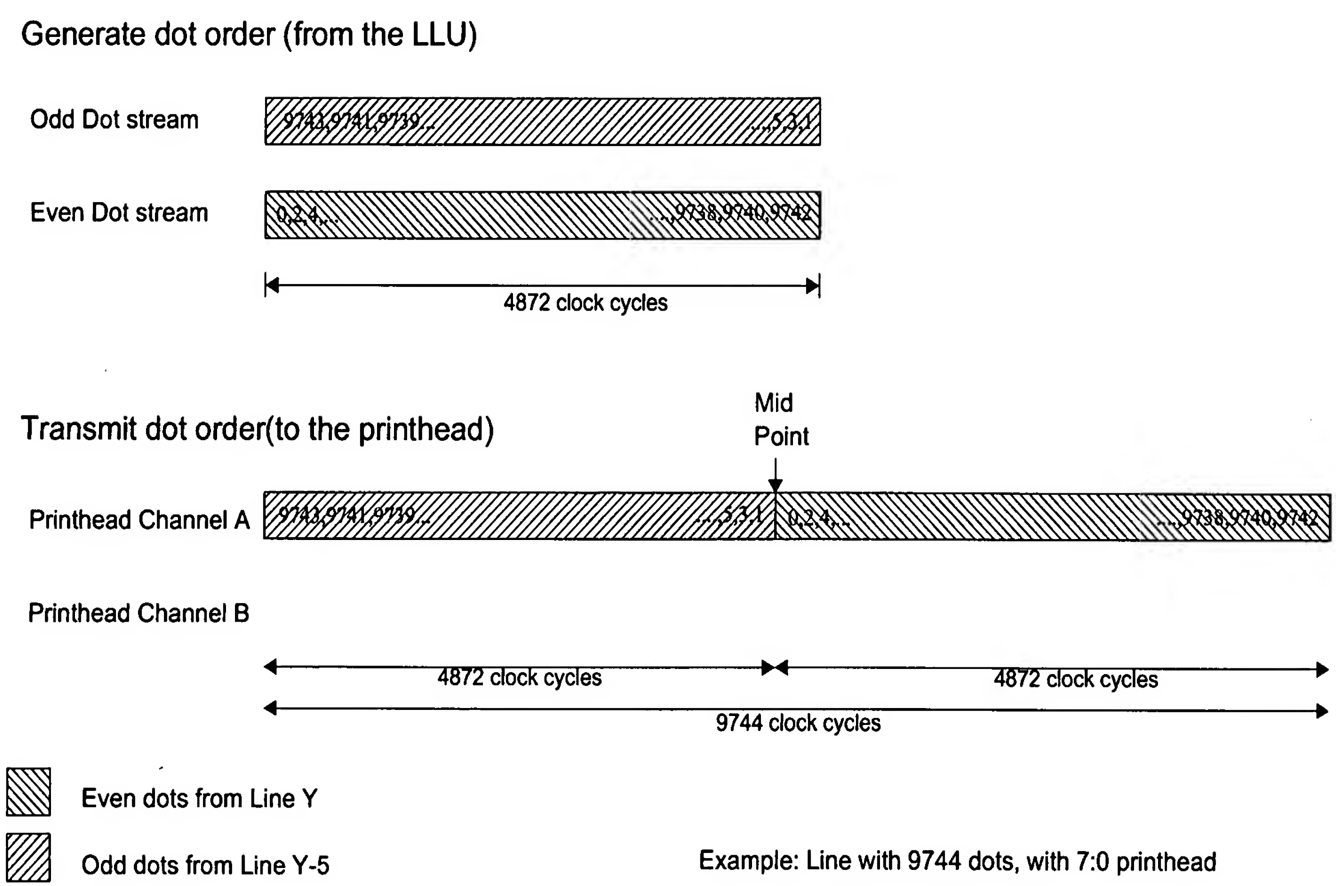

Generate dot order (from the LLU)
Odd Dot stream
9743,9741,9739...
...,5,3,1
Even Dot stream
0,2,4,...
...,9738,9740,9742
4872 clock cycles
Transmit dot order(to the printhead)
Mid
Point
Printhead Channel A
9743,9741,9739...
...,5,3,1
0,2,4,...
...,9738,9740,9742
Printhead Channel B
4872 clock cycles
4872 clock cycles
9744 clock cycles
Even dots from Line Y
Odd dots from Line Y-5
Example: Line with 9744 dots, with 7:0 printhead

FIG. 282

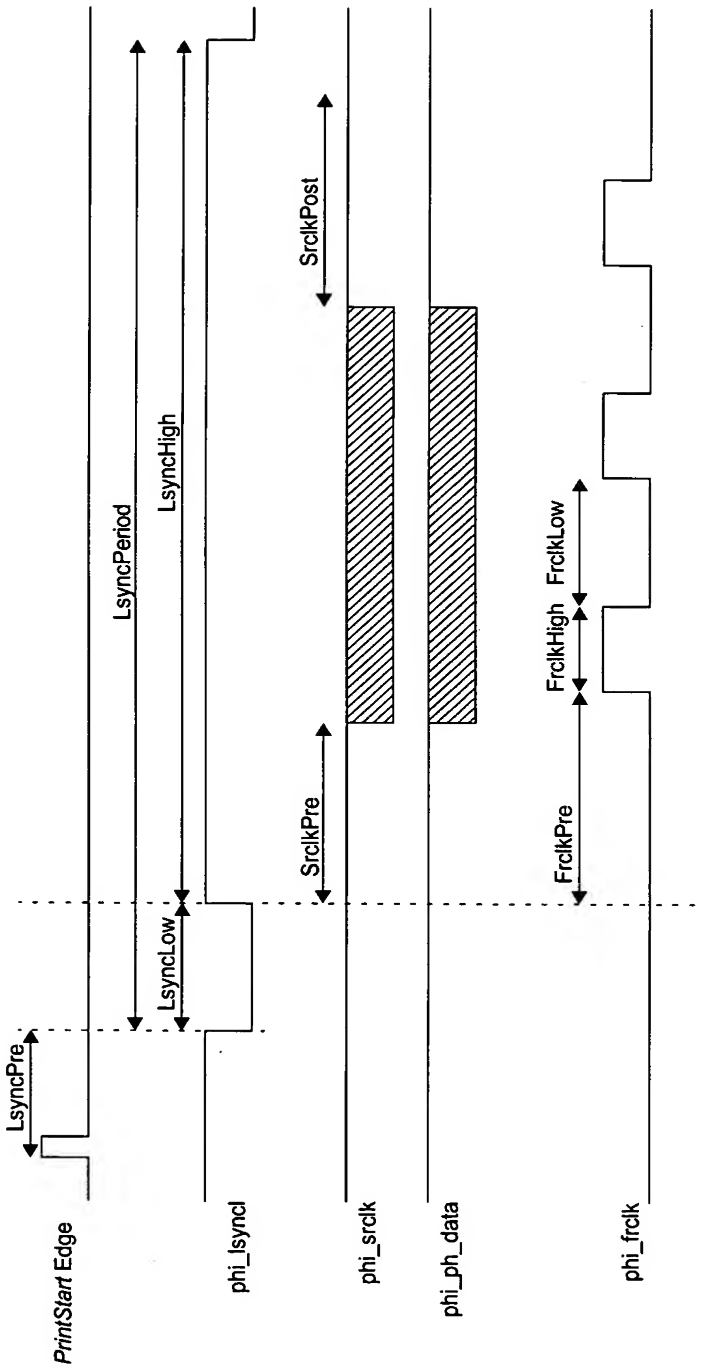
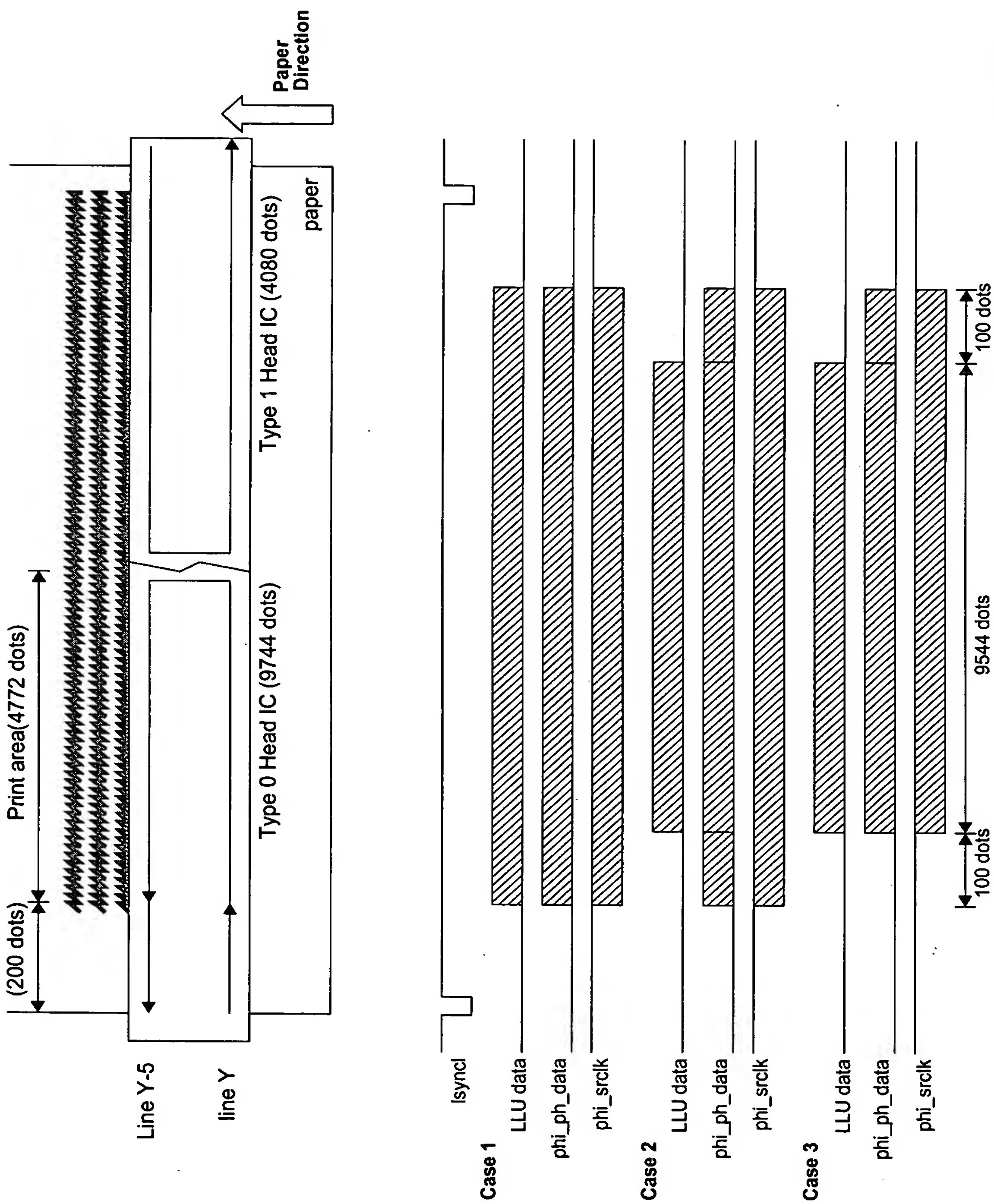

PrintStart Edge
phi_lsyncl
phi_srclk
phi_ph_data
phi_frclk
LsyncPre
LsyncLow
LsyncPeriod
LsyncHigh
SrclkPre
SrclkPost
FrclkPre
FrclkHigh
FrclkLow

FIG. 283



Print area(4772 dots)
(200 dots)
Type 0 Head IC (9744 dots)
Type 1 Head IC (4080 dots)
paper
Paper Direction
Line Y-5
line Y
lsyncl
Case 1
LLU data
phi_ph_data
phi_srclk
Case 2
LLU data
phi_ph_data
phi_srclk
Case 3
LLU data
phi_ph_data
phi_srclk
100 dots
9544 dots
100 dots

FIG. 284

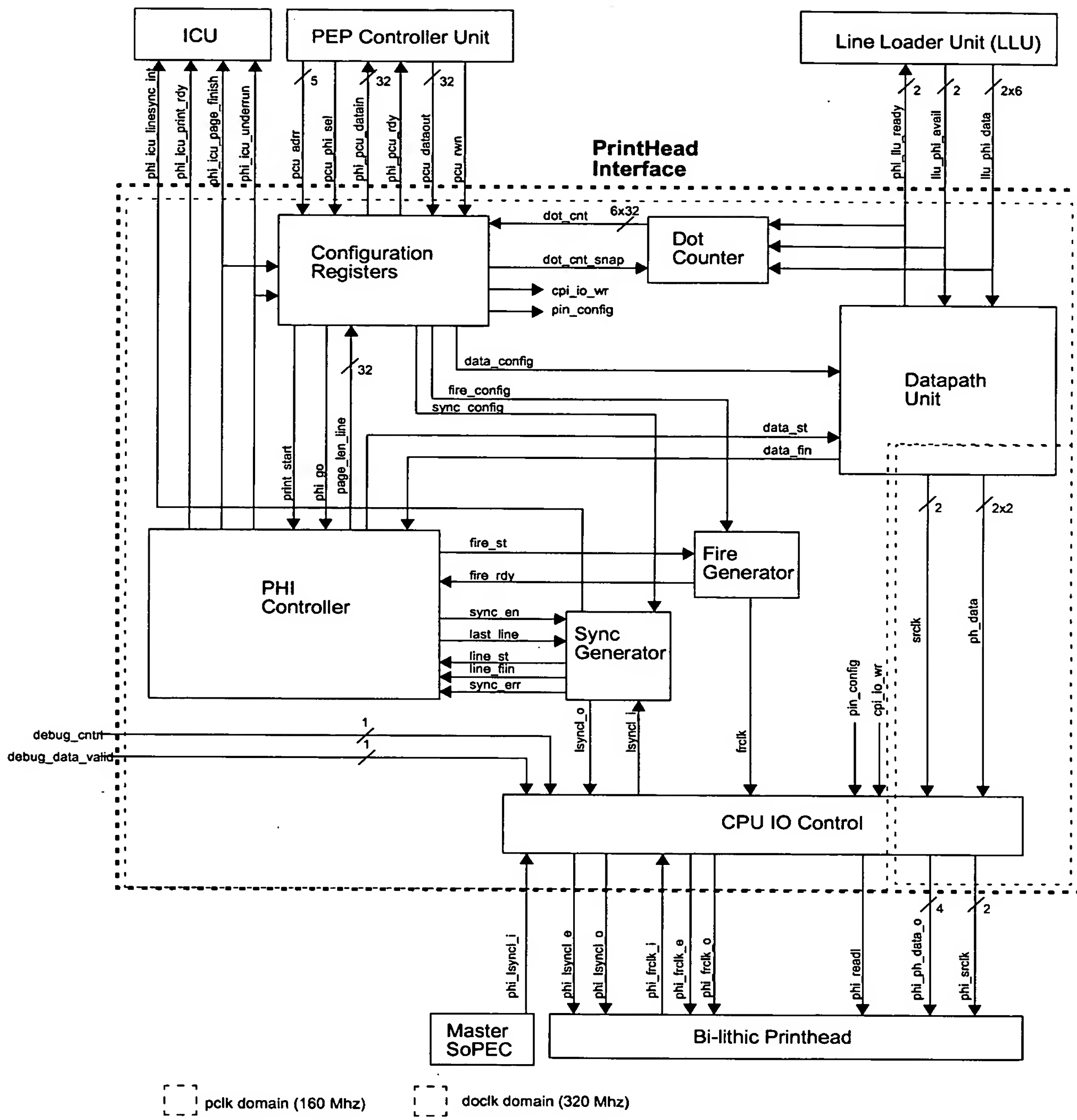
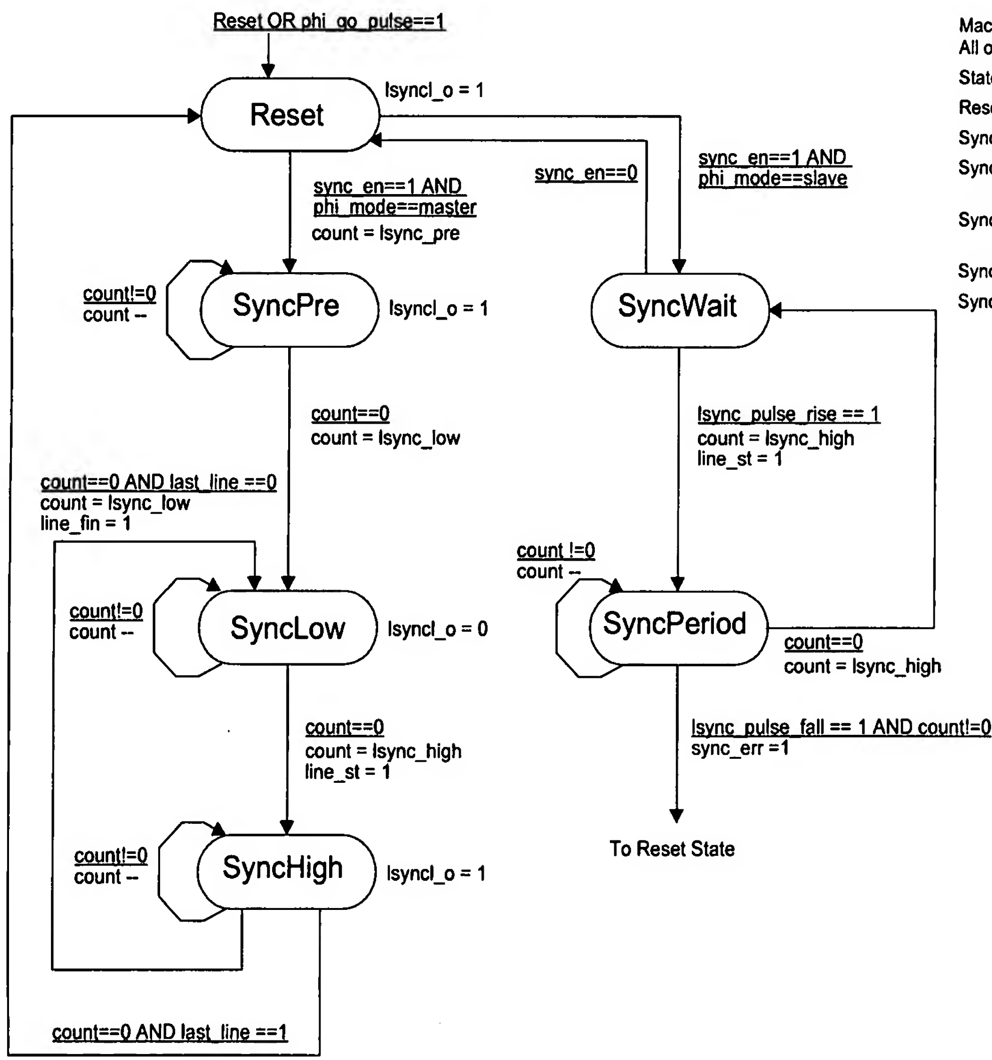

ICU
PEP Controller Unit
Line Loader Unit (LLU)
PrintHead Interface
phi_icu_linesync_int
phi_icu_print_rdy
phi_icu_page_finish
phi_icu_underrun
pcu_adrr
pcu_phi_sel
phi_pcu_datain
phi_pcu_rdy
pcu_dataout
pcu_rwn
5
32
32
phi_llu_ready
llu_phi_avail
llu_phi_data
2
2
2x6
Configuration Registers
dot_cnt
6x32
dot_cnt_snap
Dot Counter
cpi_io_wr
pin_config
data_config
fire_config
sync_config
Datapath Unit
data_st
data_fin
32
print_start
phi_go
page_len_line
PHI Controller
fire_st
fire_rdy
Fire Generator
sync_en
last_line
line_st
line_fin
sync_err
Sync Generator
2
2x2
srclk
ph_data
pin_config
cpi_io_wr
debug_cntrl
debug_data_valid
1
1
lsyncl_o
lsyncl_i
frclk
CPU IO Control
4
2
phi_lsyncl_i
phi_lsyncl_e
phi_lsyncl_o
phi_frclk_i
phi_frclk_e
phi_frclk_o
phi_readl
phi_ph_data_o
phi_srclk
Master SoPEC
Bi-lithic Printhead
pclk domain (160 Mhz)
doclk domain (320 Mhz)

FIG. 285



Machine remains in same state by default
All outputs are zero unless otherwise stated

State Description:

Reset:	Normal reset state
SyncPre:	Count the LsyncPre number of clock cycles
SyncLow:	Count the LsyncLow number of clock cycles
SyncHigh:	Count the LsyncHigh number of clock cycles
SyncWait:	Wait for an input lsync pulse
SyncPeriod:	Count the LsyncMinperiod number of clock cycles

FIG. 286

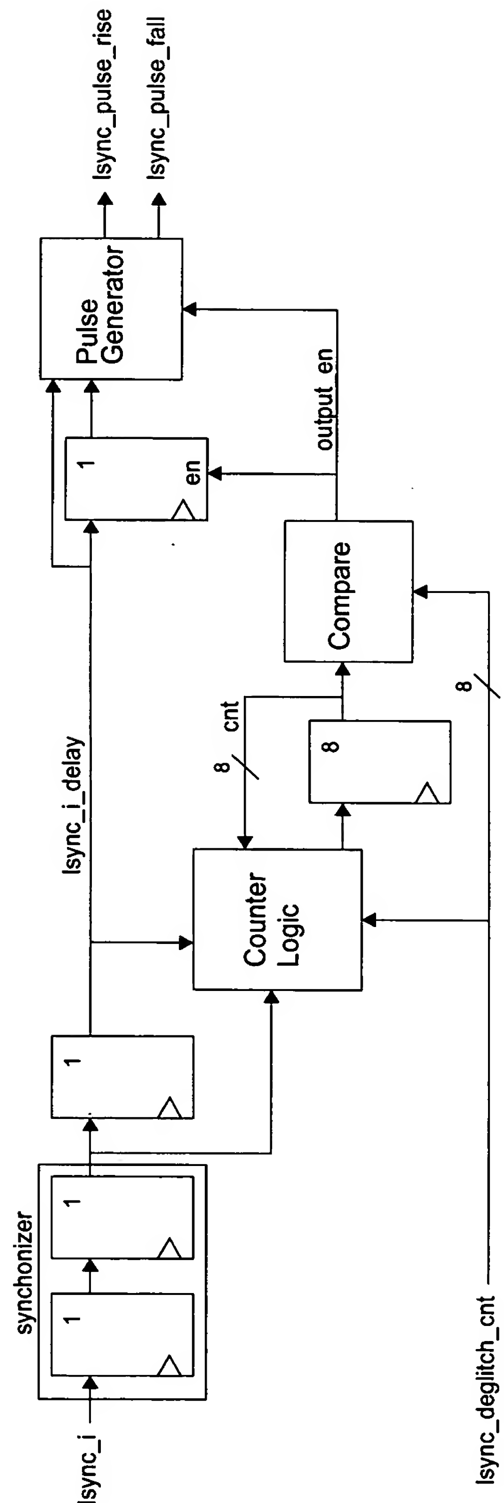
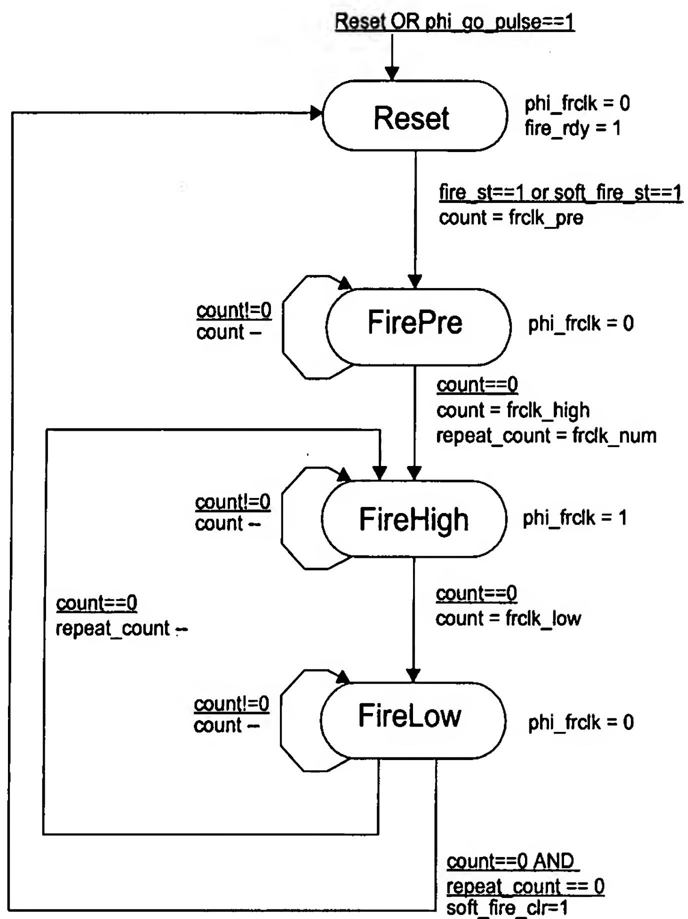

lsync_pulse_rise
lsync_pulse_fall
Pulse Generator
1
en
output_en
Compare
8
cnt
8
8
lsync_i_delay
Counter Logic
1
synchonizer
1
1
lsync_i
lsync_deglitch_cnt

FIG. 287

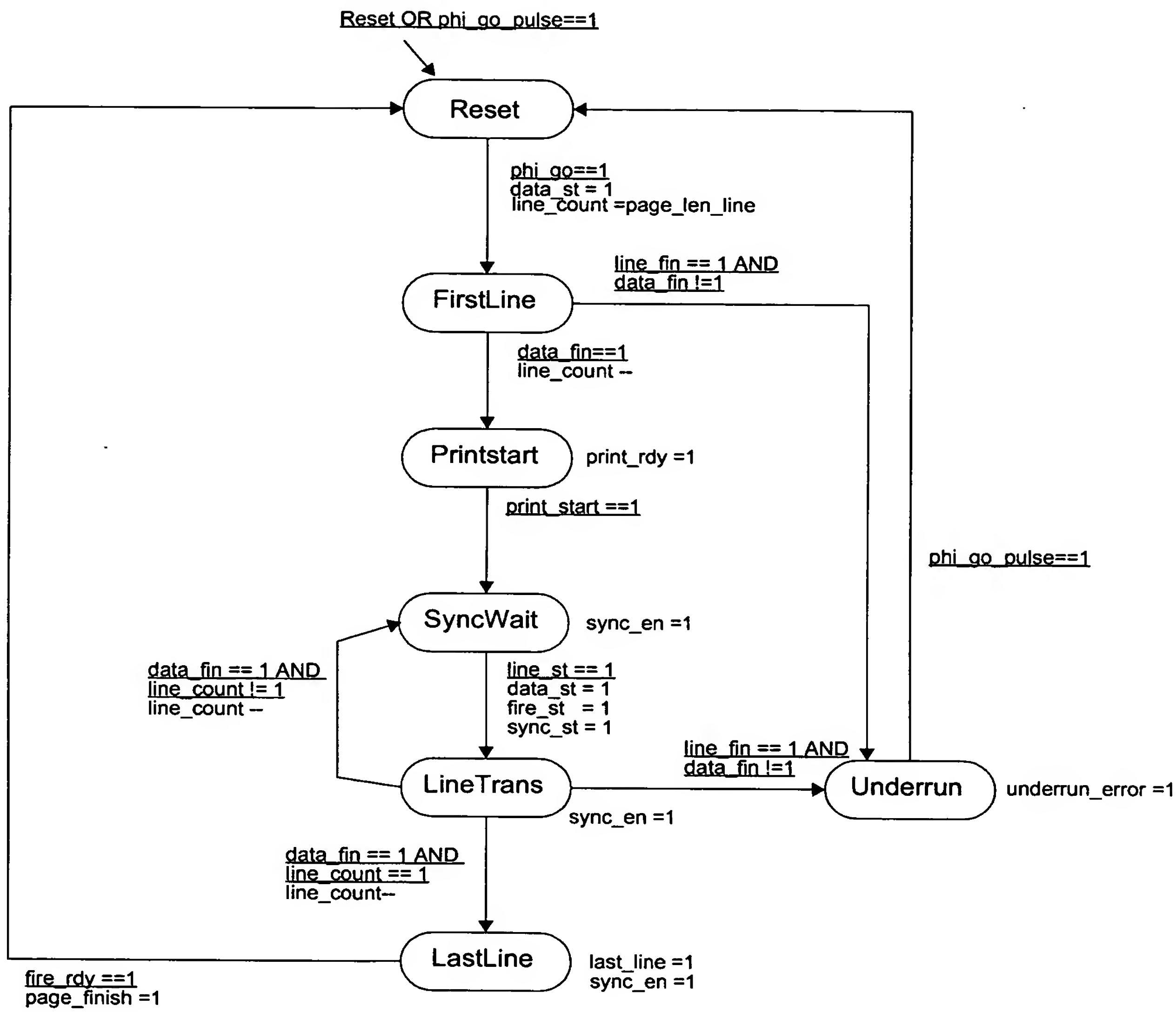


Machine remains in same state by default
All outputs are zero unless otherwise stated

State Description:

Reset:	Normal reset state
FirePre:	Count the FrclkPre number of clock cycles, repeat count set to FrclkNum
FireHigh:	Count the FrclkHigh number of clock cycles
FireLow:	Count the FrclkLow number of clock cycles

FIG. 288



Reset OR phi_go_pulse==1
Reset
phi_go==1
data_st = 1
line_count =page_len_line
FirstLine
line_fin == 1 AND
data_fin !=1
data_fin==1
line_count --
Printstart
print_rdy =1
print_start ==1
phi_go_pulse==1
SyncWait
sync_en =1
data_fin == 1 AND
line_count != 1
line_count --
line_st == 1
data_st = 1
fire_st = 1
sync_st = 1
line_fin == 1 AND
data_fin !=1
LineTrans
sync_en =1
Underrun
underrun_error =1
data_fin == 1 AND
line_count == 1
line_count--
LastLine
last_line =1
sync_en =1
fire_rdy ==1
page_finish =1

FIG. 289

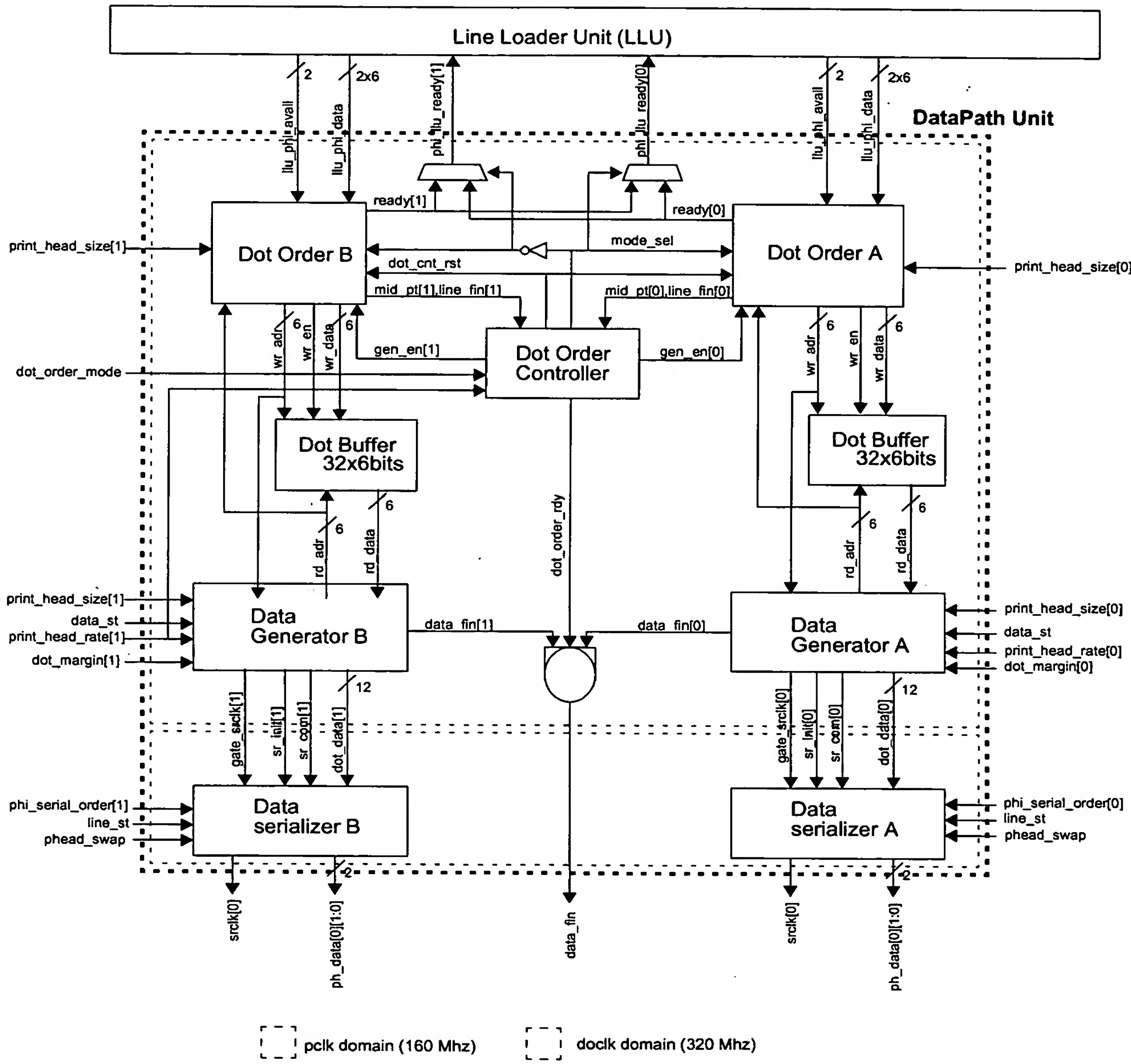
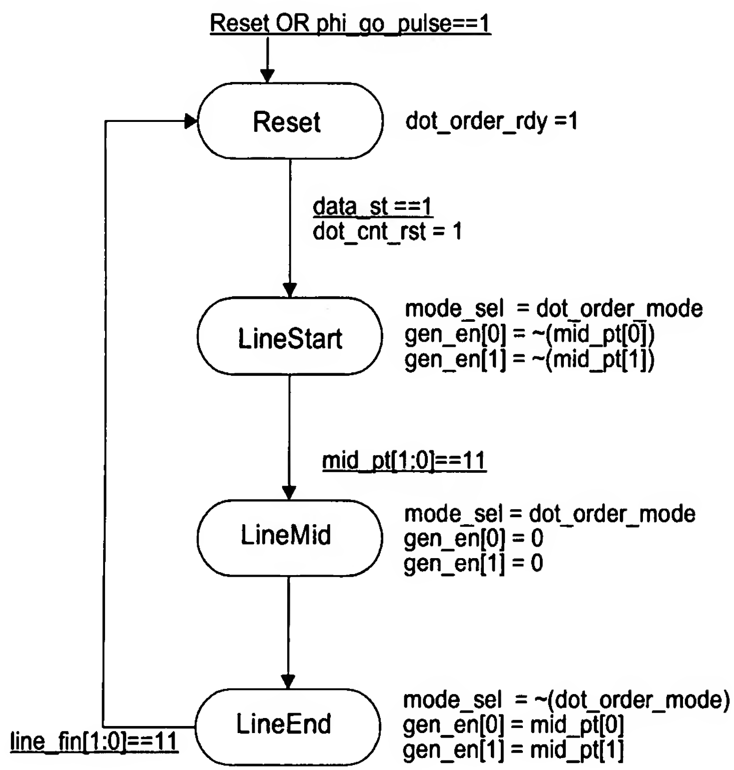

Line Loader Unit (LLU)
DataPath Unit
llu_phi_avail
llu_phi_data
phi_llu_ready[1]
phi_llu_ready[0]
2
2x6
ready[1]
ready[0]
mode_sel
dot_cnt_rst
mid_pt[1],line_fin[1]
mid_pt[0],line_fin[0]
Dot Order B
Dot Order A
print_head_size[1]
print_head_size[0]
wr_adr
wr_en
wr_data
6
gen_en[1]
gen_en[0]
Dot Order Controller
dot_order_mode
Dot Buffer 32x6bits
rd_adr
rd_data
dot_order_rdy
Data Generator B
Data Generator A
data_st
print_head_rate[1]
print_head_rate[0]
dot_margin[1]
dot_margin[0]
data_fin[1]
data_fin[0]
12
gate_sclk[1]
sr_init[1]
sr_com[1]
dot_data[1]
gate_srclk[0]
sr_init[0]
sr_com[0]
dot_data[0]
Data serializer B
Data serializer A
phi_serial_order[1]
phi_serial_order[0]
line_st
phead_swap
srclk[0]
ph_data[0][1:0]
data_fin
pclk domain (160 Mhz)
doclk domain (320 Mhz)

FIG. 290

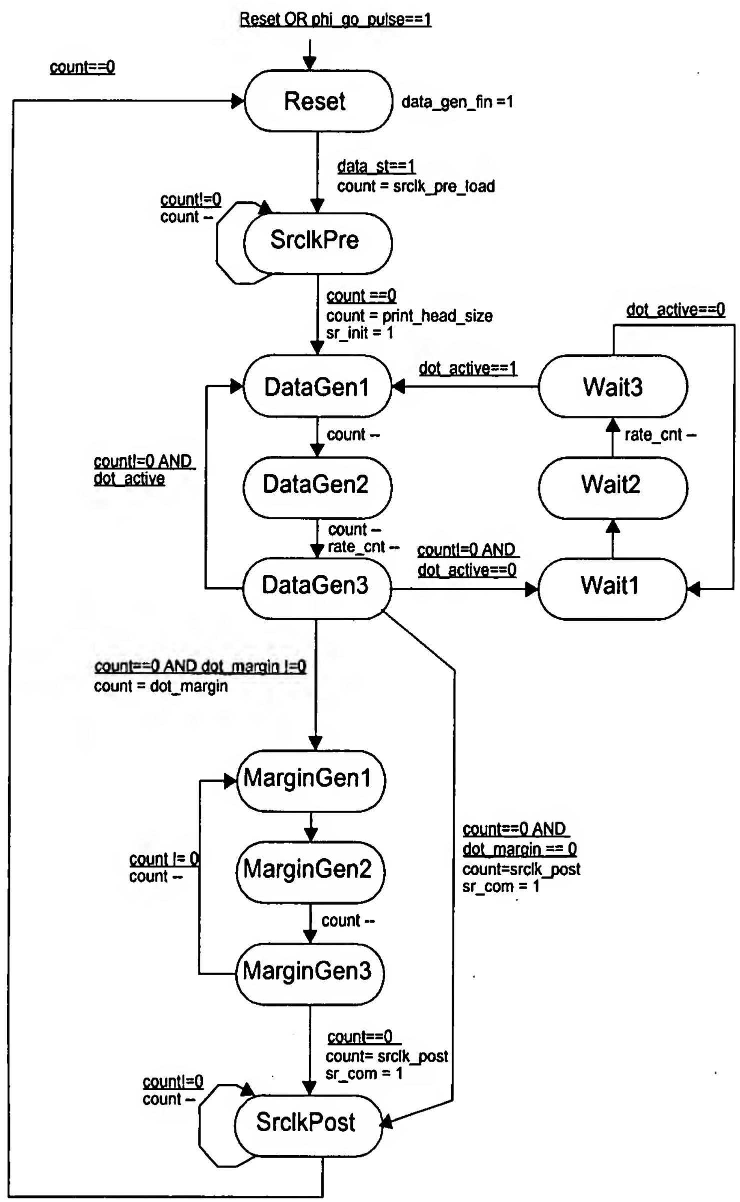


Machine remains in same state by default
All outputs are zero unless otherwise stated

State Description:

Reset:	Normal reset state
Linestart:	Start processing first part of the line, wait for both mid_pt to be active
LineMid:	Switch over wait state allow pipeline to clear
LineEnd:	Line end processing wait for both line_fin to be active

FIG. 291



Machine remains in same state by default
All outputs are zero unless otherwise stated

State Description:

Reset:	Normal reset state
SrclkPre:	Count the SrclkPre number of clock cycles
DataGen1:	Read Line Dot data from buffer
DataGen2:	Read Line Dot data from buffer
DataGen3:	Data gen wait state
Wait1:	Data rate control wait state
Wait2:	Data rate control wait state
Wait3:	Data rate control wait state
MarginGen1:	Generate *DotMargin* number of dots
MarginGen2:	Generate *DotMargin* number of dots
MarginGen3:	Generate margin wait state
SrclkPost:	Wait for SrclkPost number of cycles

FIG. 292

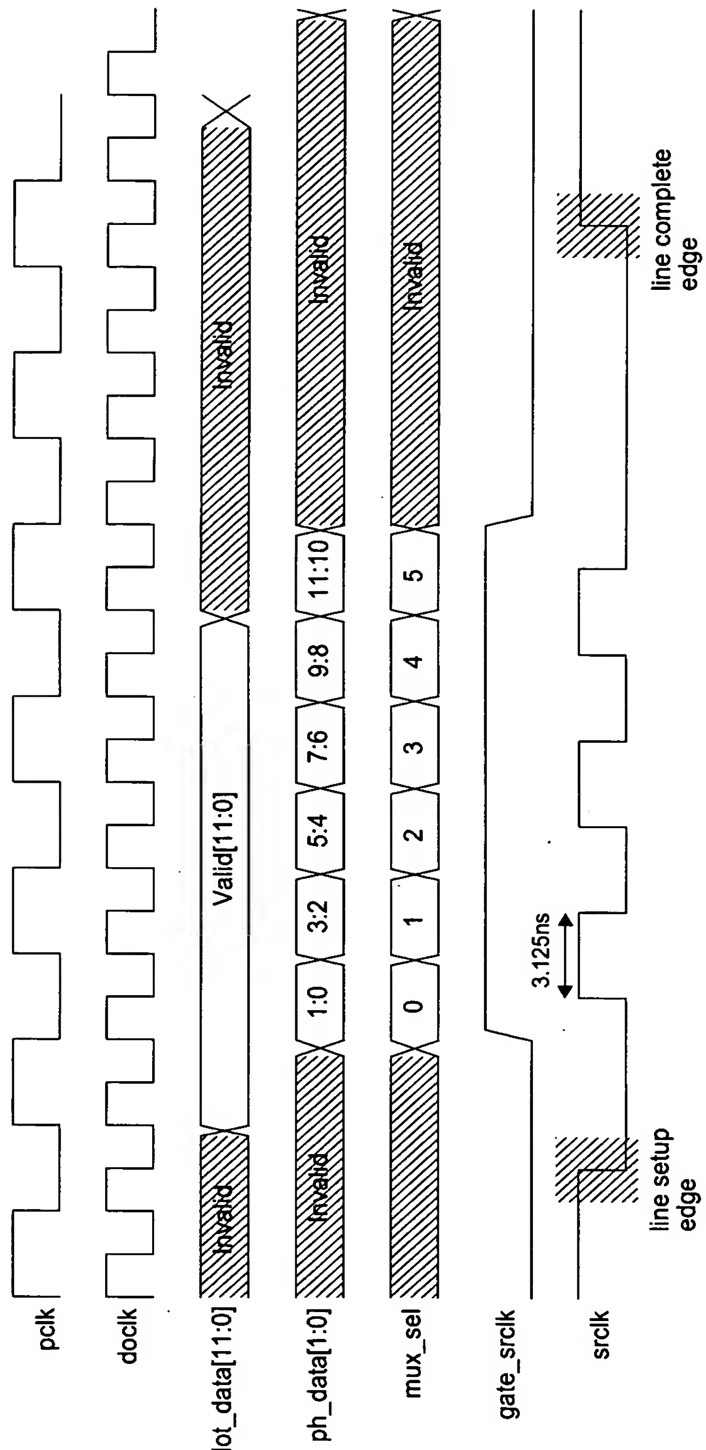
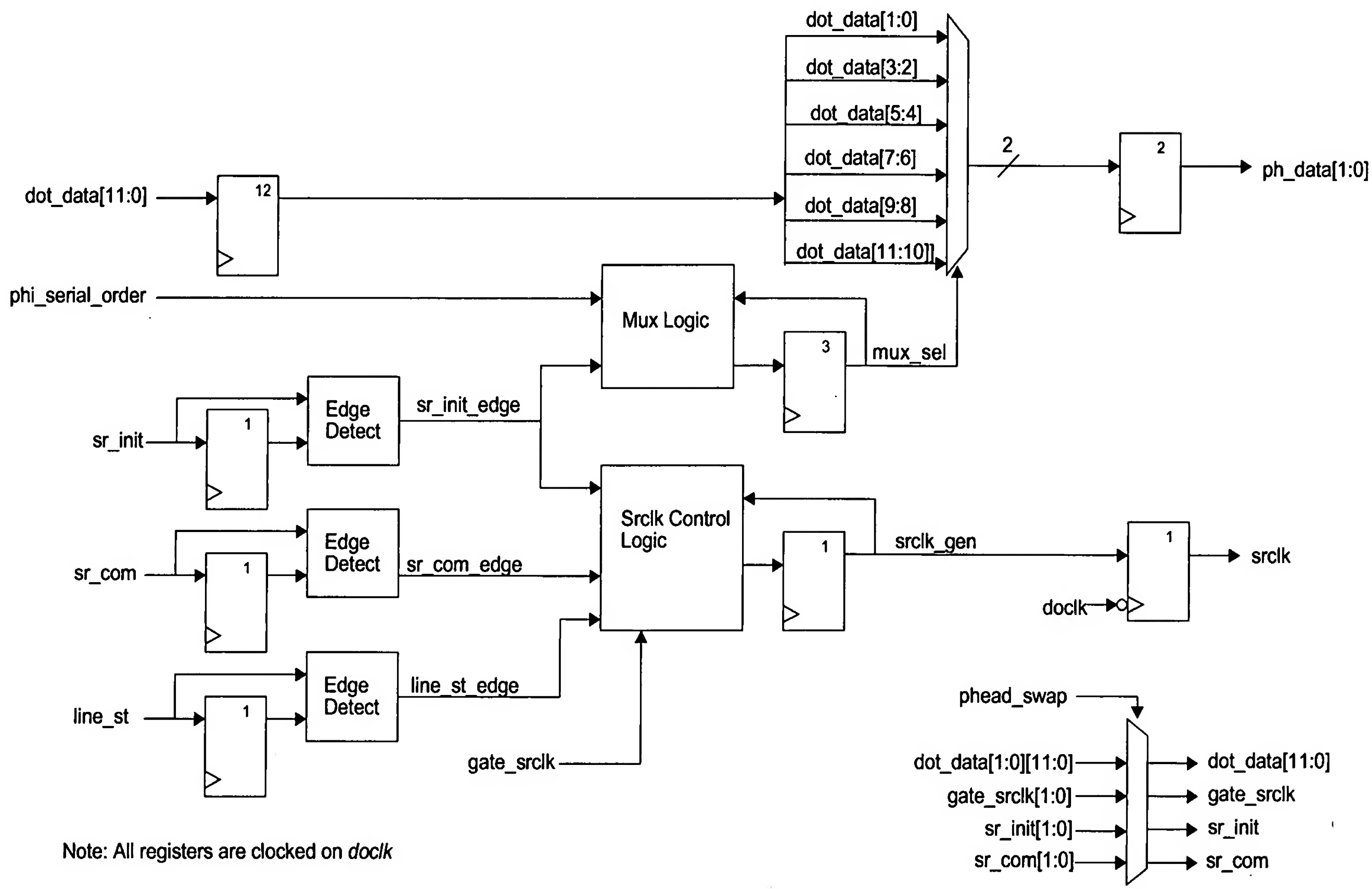

pclk
doclk
dot_data[11:0]
ph_data[1:0]
mux_sel
gate_srclk
srclk
Invalid
Valid[11:0]
Invalid
Invalid
1:0
3:2
5:4
7:6
9:8
11:10
Invalid
Invalid
0
1
2
3
4
5
Invalid
3.125ns
line setup edge
line complete edge

FIG. 293



dot_data[11:0]
12
phi_serial_order
sr_init
1
Edge Detect
sr_init_edge
sr_com
1
Edge Detect
sr_com_edge
line_st
1
Edge Detect
line_st_edge
gate_srclk
Mux Logic
3
mux_sel
Srclk Control Logic
1
srclk_gen
dot_data[1:0]
dot_data[3:2]
dot_data[5:4]
dot_data[7:6]
dot_data[9:8]
dot_data[11:10]]
2
2
ph_data[1:0]
1
srclk
doclk
phead_swap
dot_data[1:0][11:0]
gate_srclk[1:0]
sr_init[1:0]
sr_com[1:0]
dot_data[11:0]
gate_srclk
sr_init
sr_com
Note: All registers are clocked on *doclk*

FIG. 294

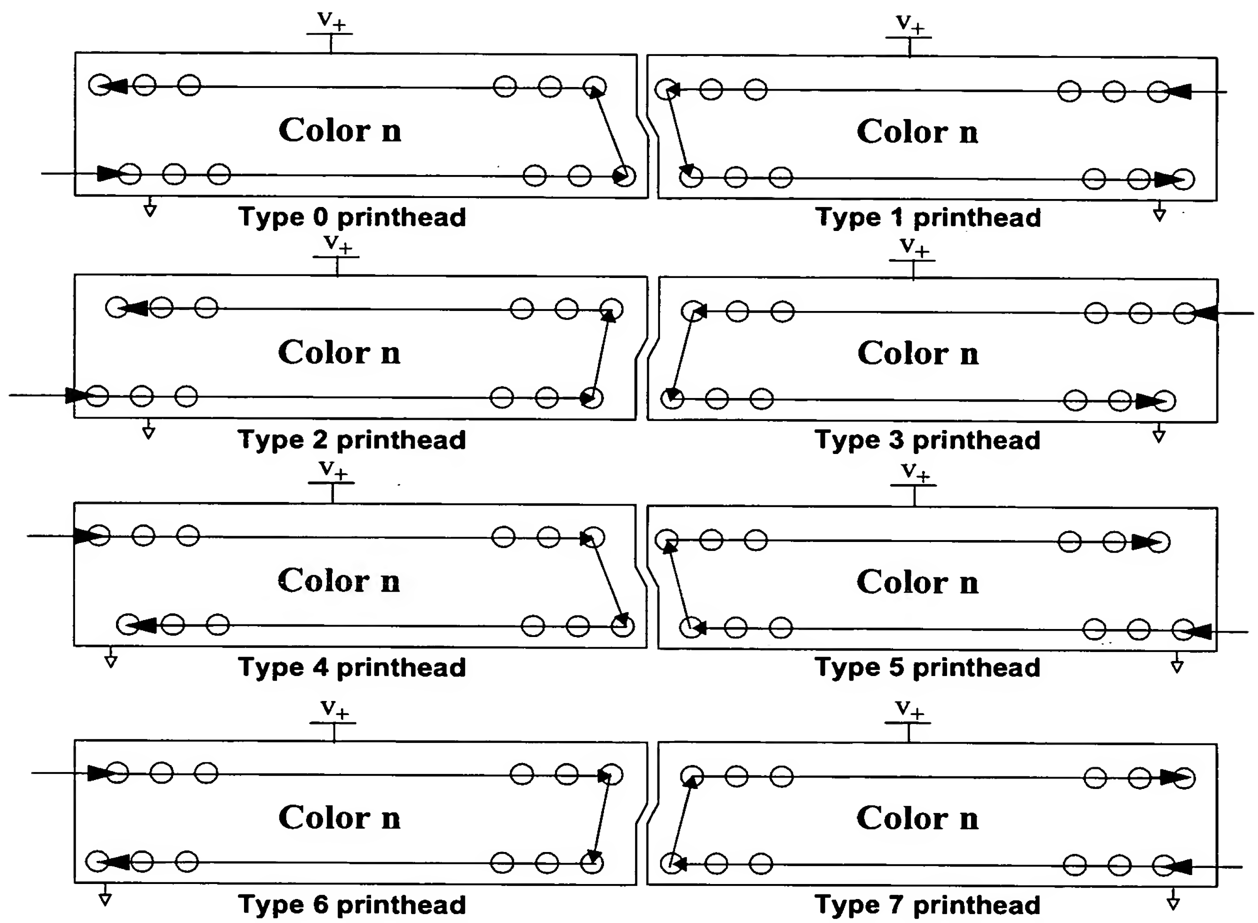

V+
Color n
Type 0 printhead
V+
Color n
Type 1 printhead
V+
Color n
Type 2 printhead
V+
Color n
Type 3 printhead
V+
Color n
Type 4 printhead
V+
Color n
Type 5 printhead
V+
Color n
Type 6 printhead
V+
Color n
Type 7 printhead

FIG. 295

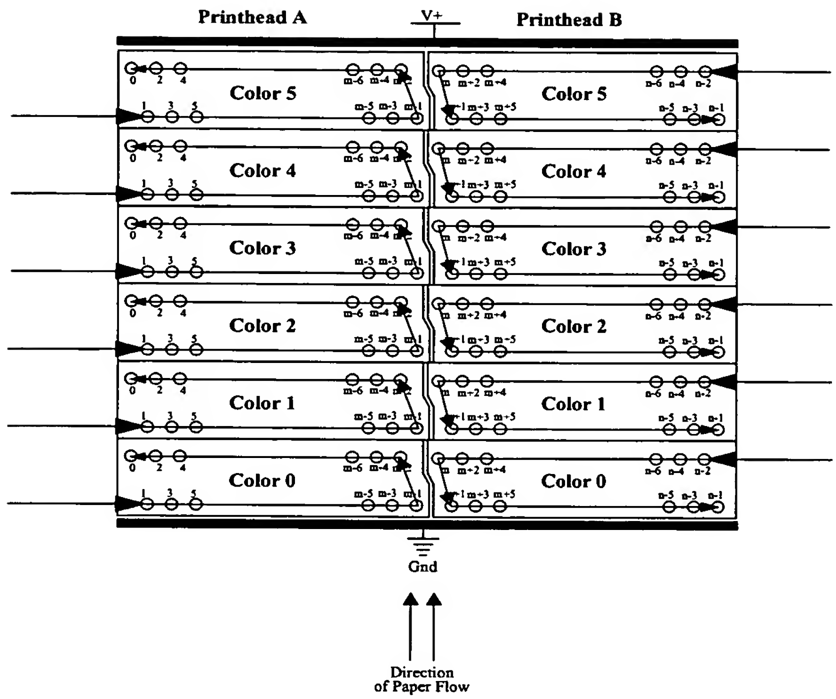
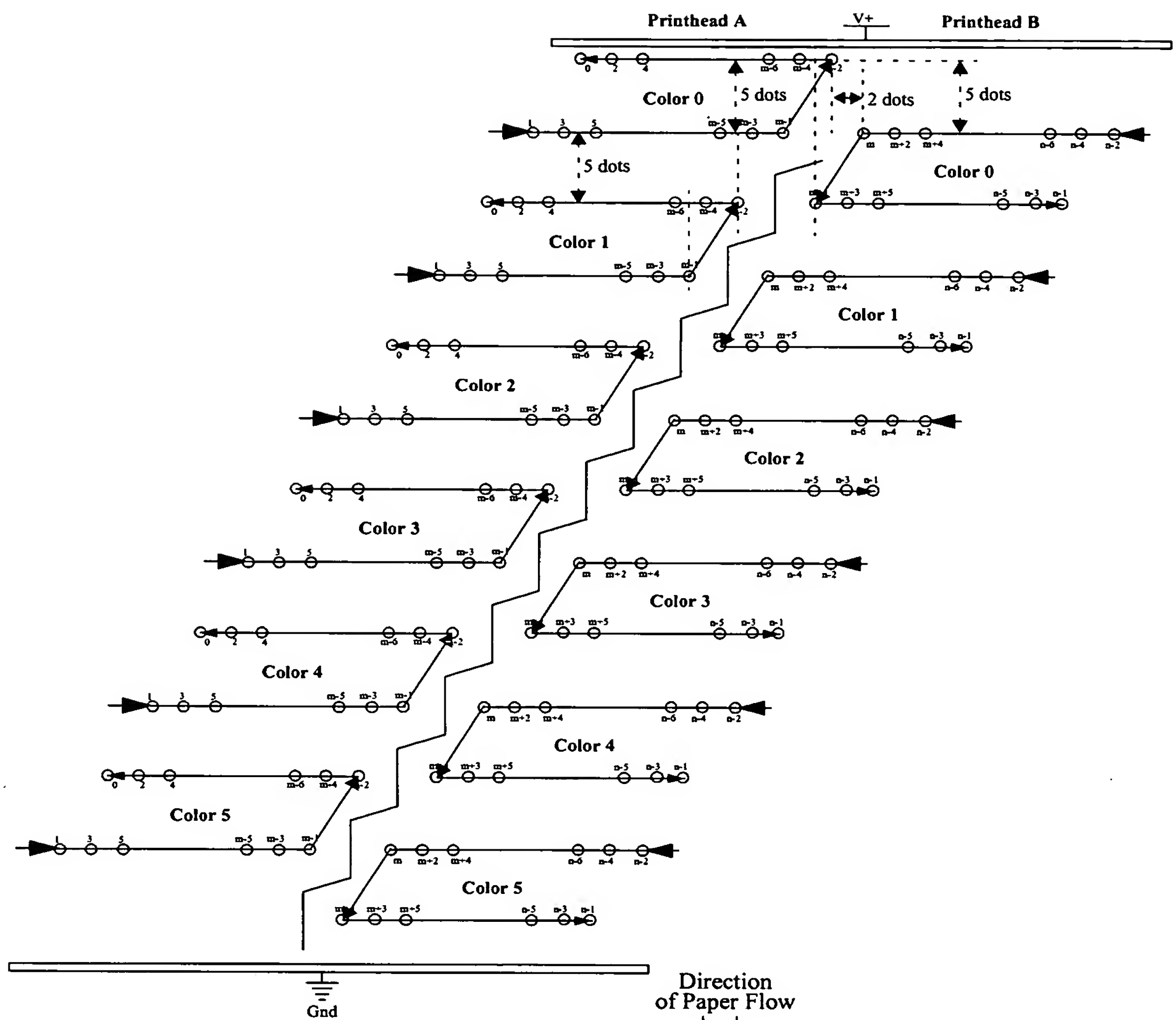

Printhead A
V+
Printhead B
Color 5
Color 4
Color 3
Color 2
Color 1
Color 0
Gnd
Direction
of Paper Flow

FIG. 296



The printheads are facing downwards.
The ink is being shot down onto the page.

FIG. 297

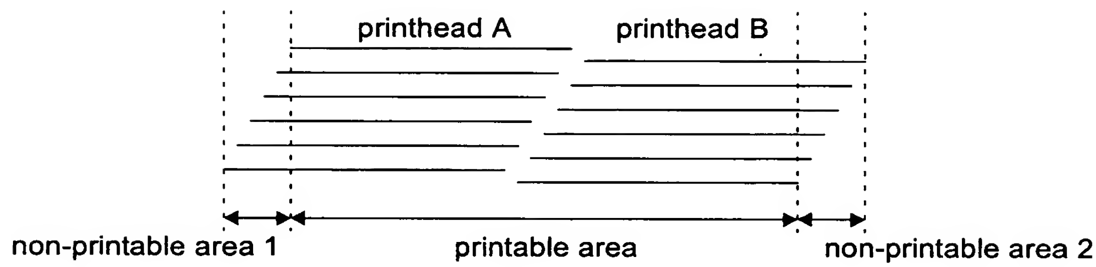
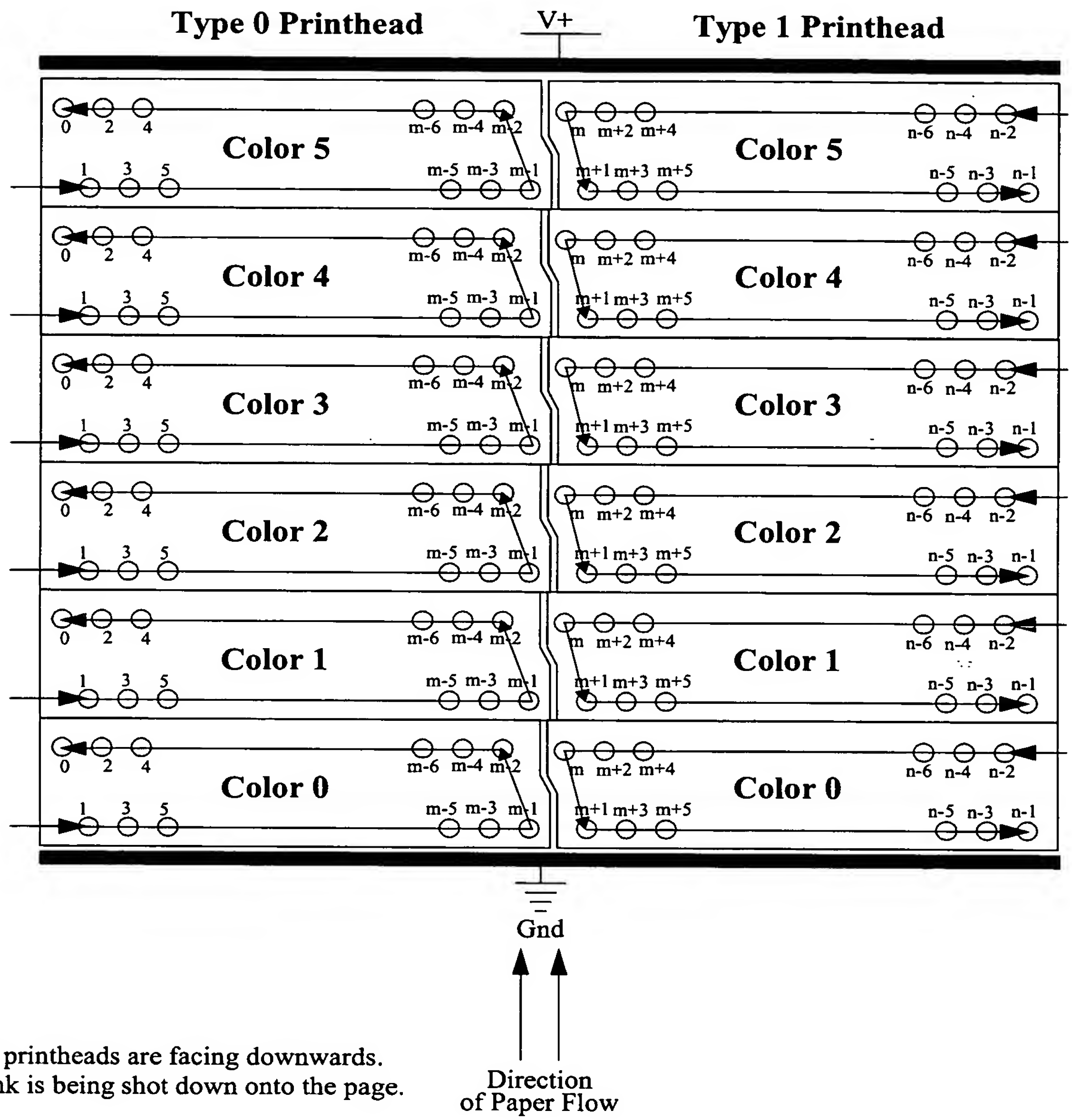

printhead A
printhead B
non-printable area 1
printable area
non-printable area 2

FIG. 298



The printheads are facing downwards.
The ink is being shot down onto the page.

FIG. 299

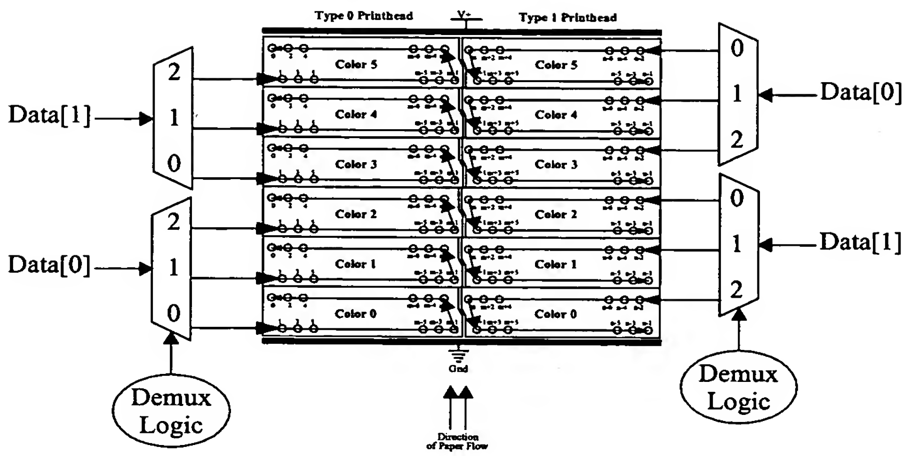


FIG. 300

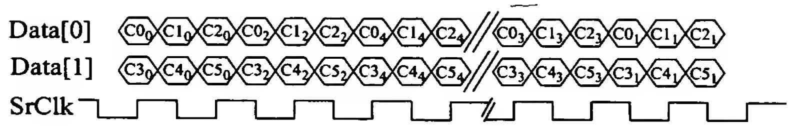


FIG. 301

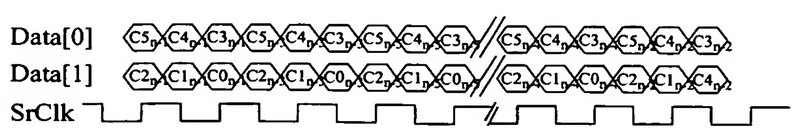


FIG. 302

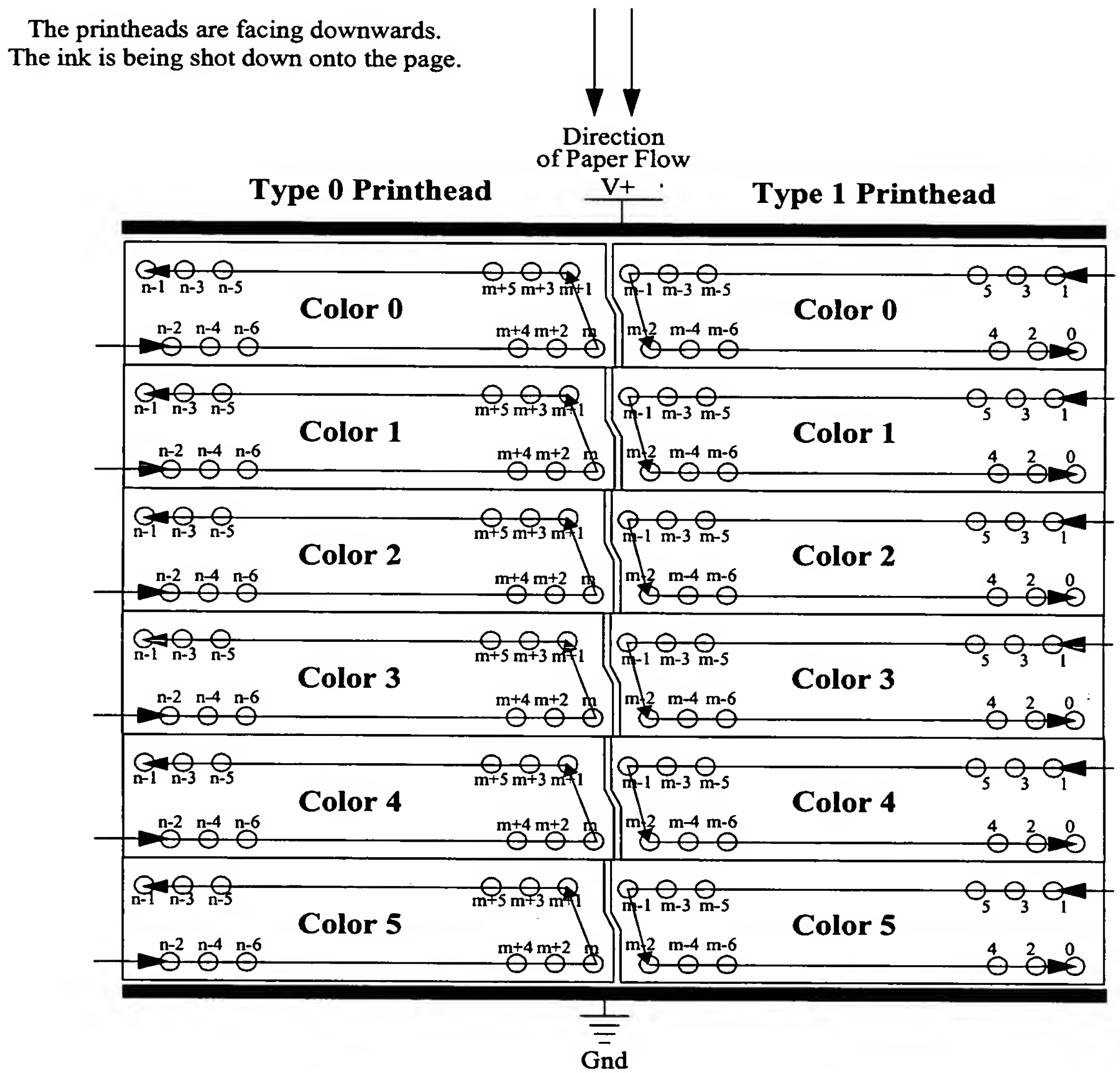

The printheads are facing downwards.
The ink is being shot down onto the page.
Direction
of Paper Flow
V+
Type 0 Printhead
Type 1 Printhead
Color 0
Color 1
Color 2
Color 3
Color 4
Color 5
n-1 n-3 n-5
n-2 n-4 n-6
m+5 m+3 m+1
m+4 m+2 m
m-1 m-3 m-5
m-2 m-4 m-6
5 3 1
4 2 0
Gnd

FIG. 303

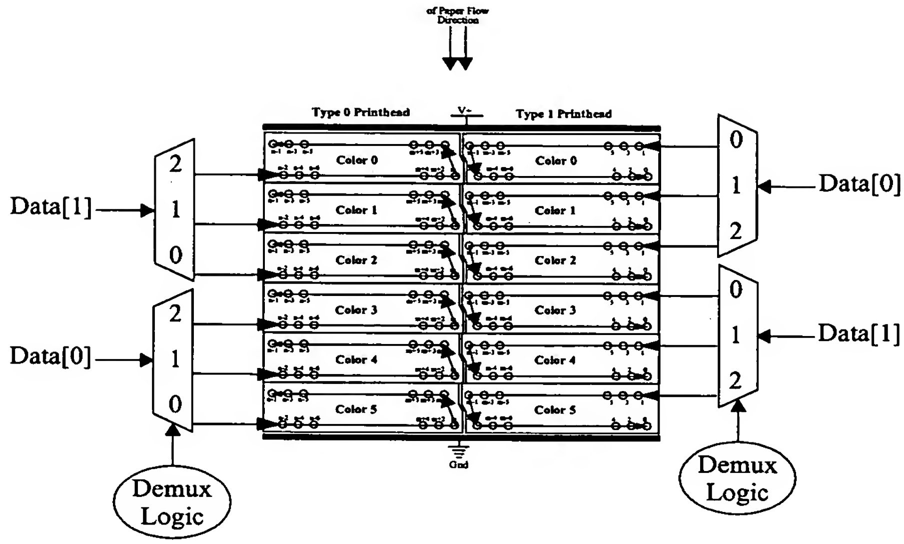


FIG. 304

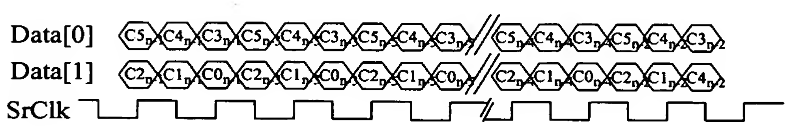


FIG. 305

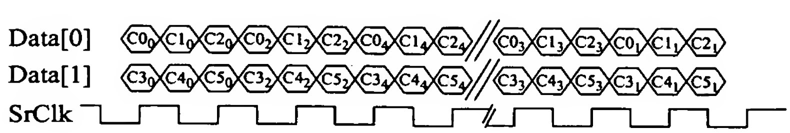


FIG. 306

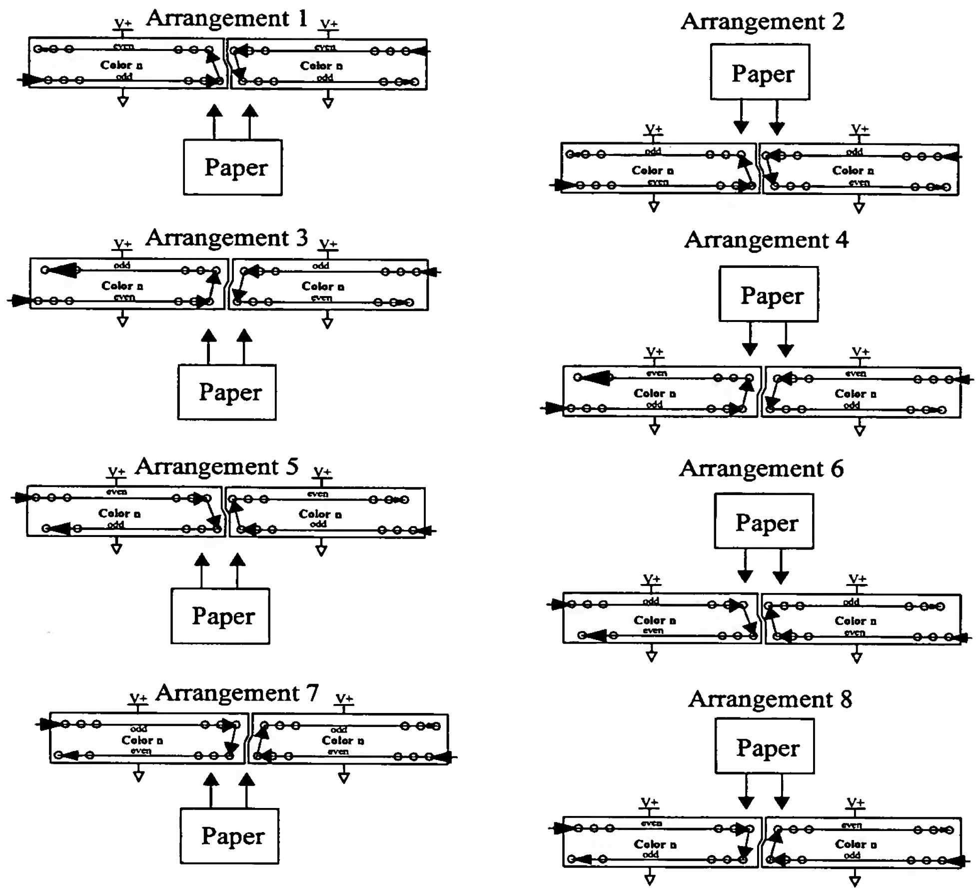

Arrangement 1
Arrangement 2
Arrangement 3
Arrangement 4
Arrangement 5
Arrangement 6
Arrangement 7
Arrangement 8
Paper
V+
even
odd
Color n

FIG. 307

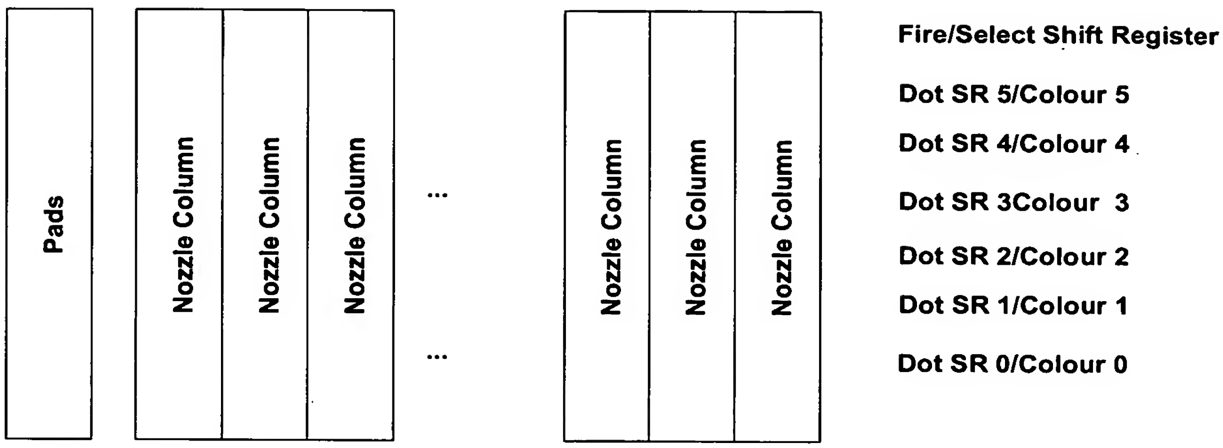

Pads
Nozzle Column
Nozzle Column
Nozzle Column
...
...
Nozzle Column
Nozzle Column
Nozzle Column
Fire/Select Shift Register
Dot SR 5/Colour 5
Dot SR 4/Colour 4
Dot SR 3Colour 3
Dot SR 2/Colour 2
Dot SR 1/Colour 1
Dot SR 0/Colour 0

FIG. 308

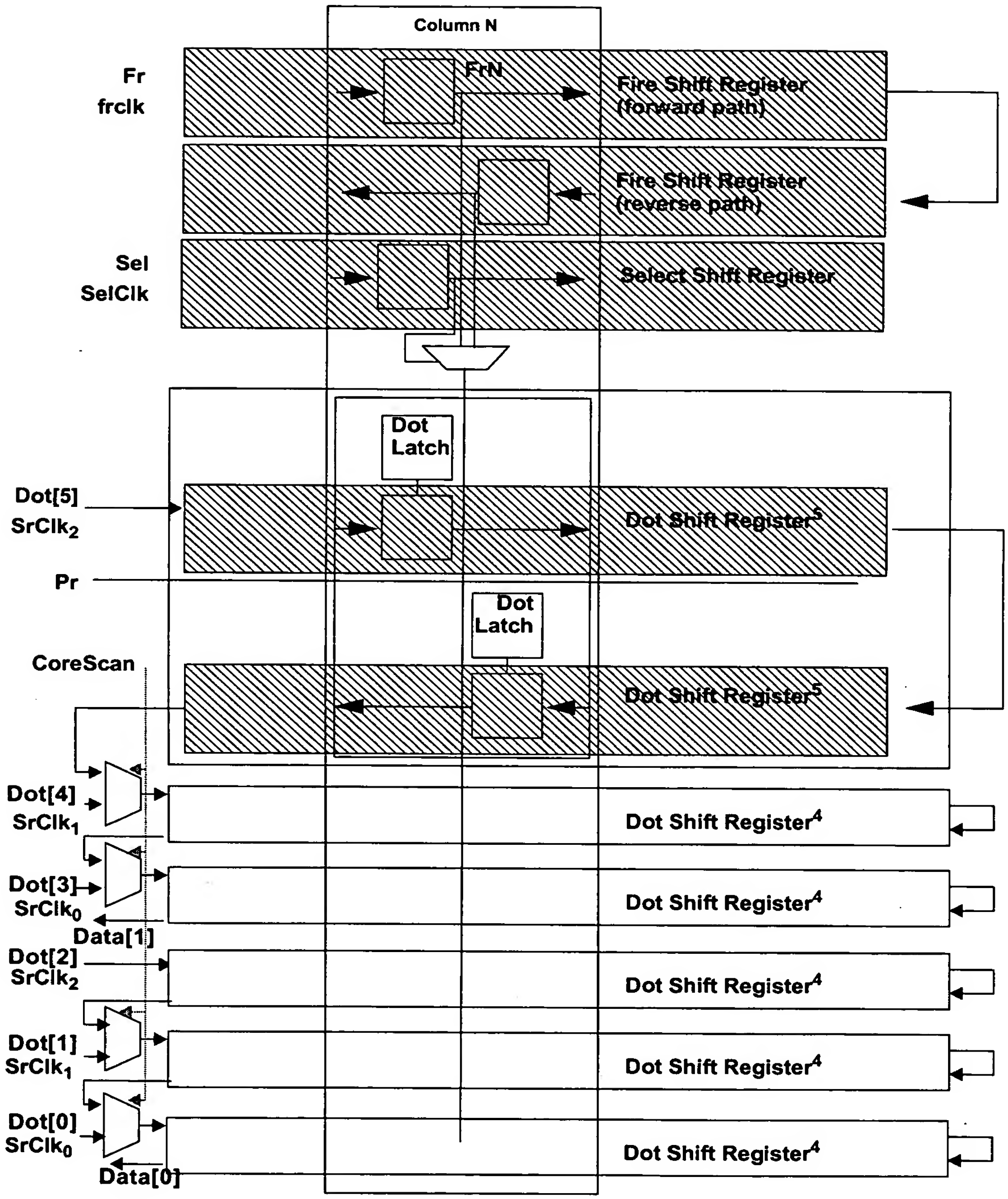

Column N
Fr
frclk
FrN
Fire Shift Register (forward path)
Fire Shift Register (reverse path)
Sel
SelClk
Select Shift Register
Dot Latch
Dot[5]
SrClk2
Dot Shift Register5
Pr
Dot Latch
CoreScan
Dot Shift Register5
Dot[4]
SrClk1
Dot Shift Register4
Dot[3]
SrClk0
Dot Shift Register4
Data[1]
Dot[2]
SrClk2
Dot Shift Register4
Dot[1]
SrClk1
Dot Shift Register4
Dot[0]
SrClk0
Dot Shift Register4
Data[0]

FIG. 309

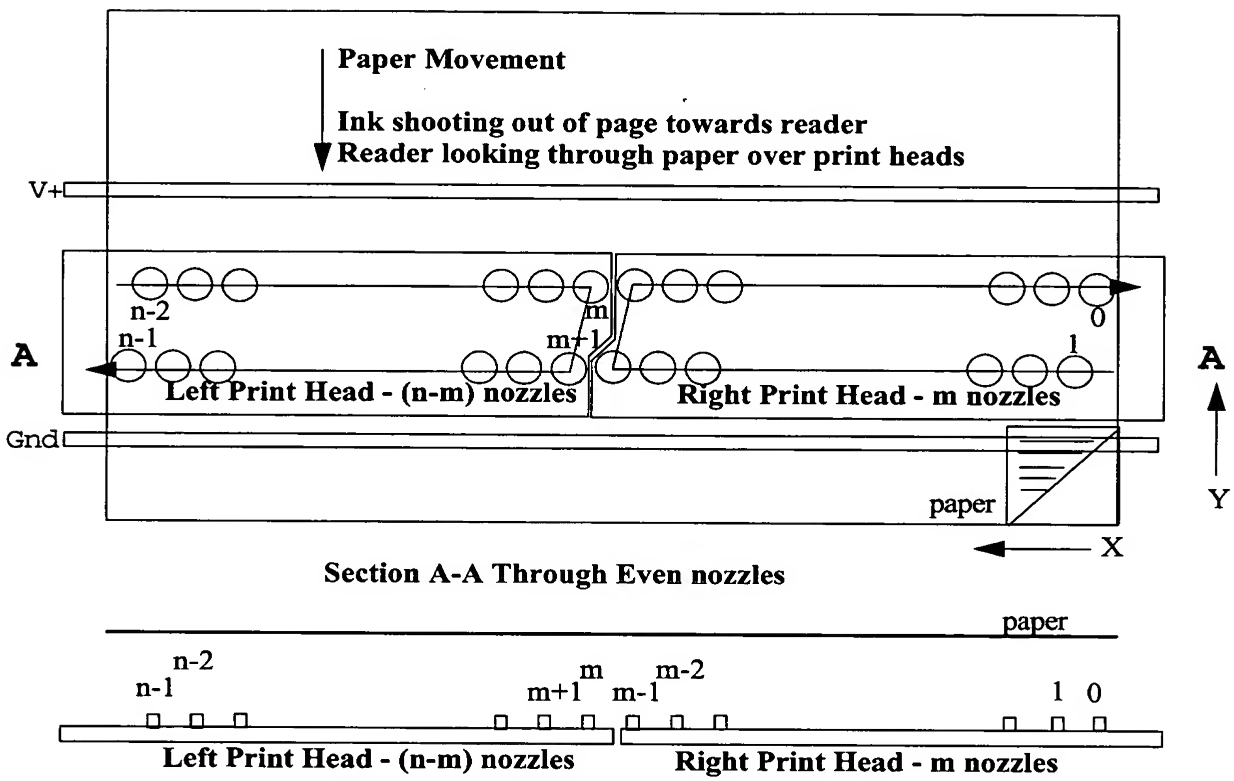
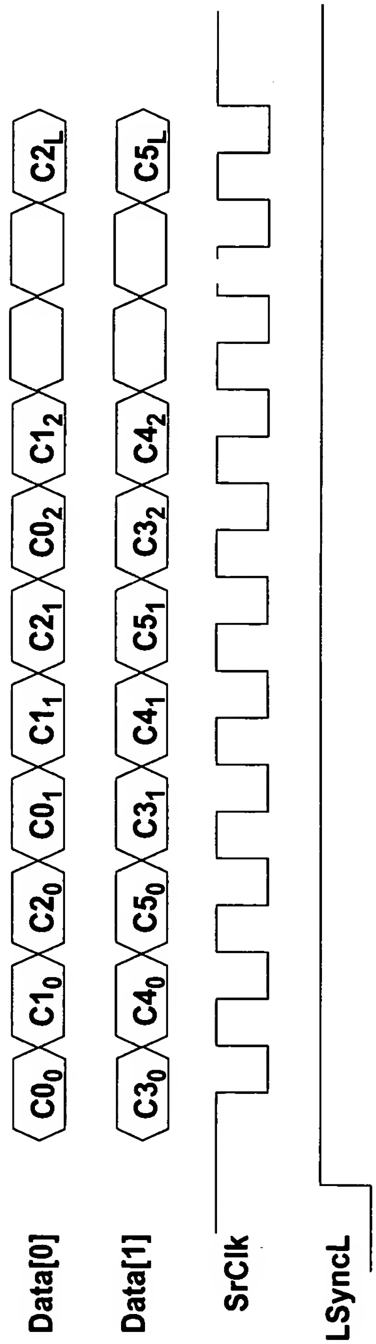
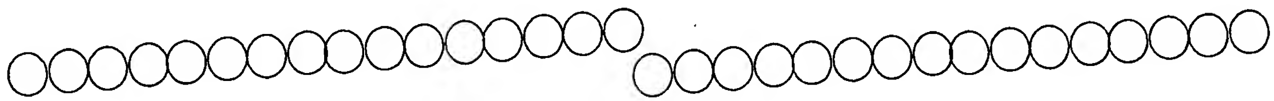

Paper Movement
Ink shooting out of page towards reader
Reader looking through paper over print heads
V+
A
n-2
n-1
m
m+1
0
1
Left Print Head - (n-m) nozzles
Right Print Head - m nozzles
A
Gnd
paper
Y
X
Section A-A Through Even nozzles
paper
n-2
n-1
m
m+1
m-1
m-2
1
0
Left Print Head - (n-m) nozzles
Right Print Head - m nozzles

FIG. 310

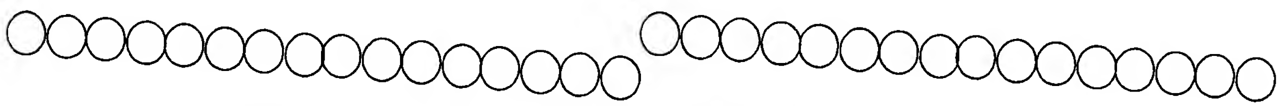


Data[0]
Data[1]
SrClk
LSyncL
C0₀ C1₀ C2₀ C0₁ C1₁ C2₁ C0₂ C1₂ C2L
C3₀ C4₀ C5₀ C3₁ C4₁ C5₁ C3₂ C4₂ C5L

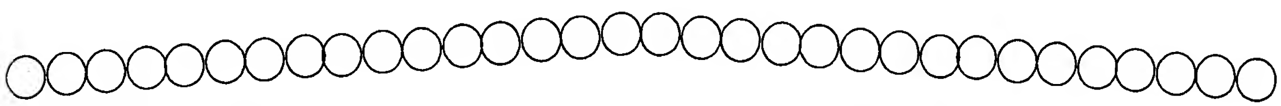
FIG. 311



a) Printing every n^{th} dot with all zero's in the fire select shift register



b) Printing every n^{th} dot with all one's in the fire select shift register



c) Printing every n^{th} dot with n zero's then n one's in the fire select shift registers

FIG. 312

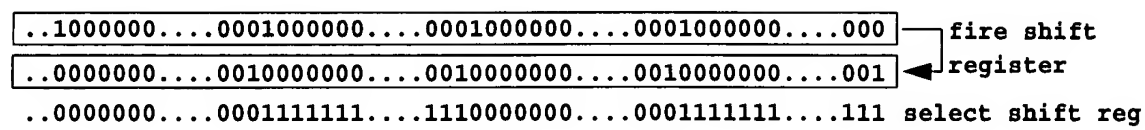


FIG. 313

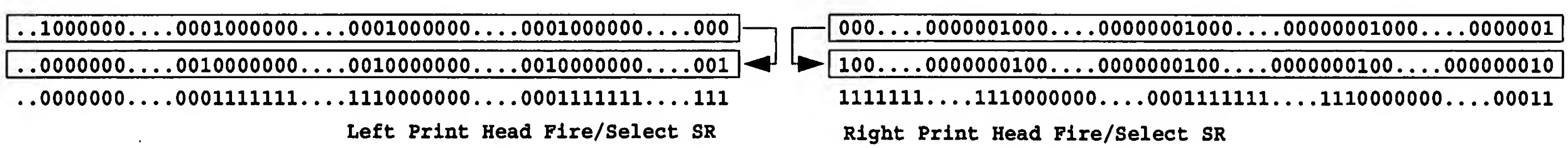


FIG. 314

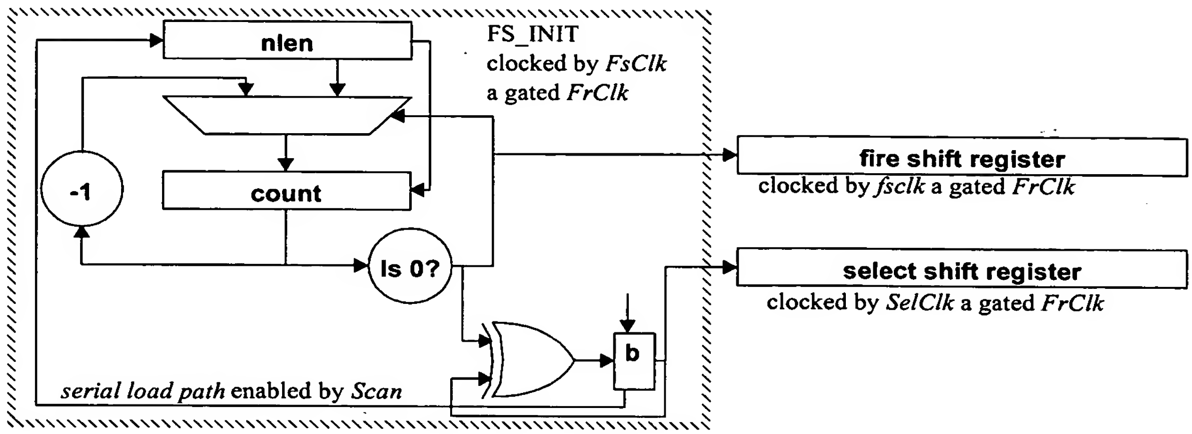


FIG. 315

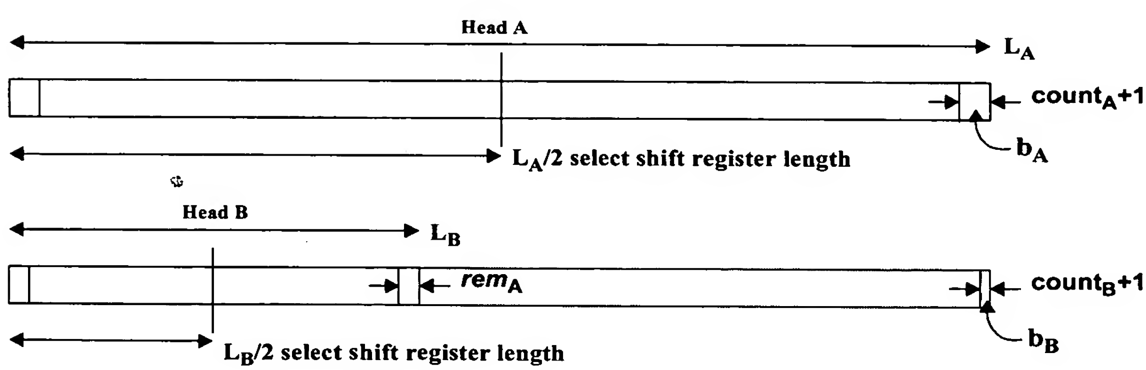


FIG. 316

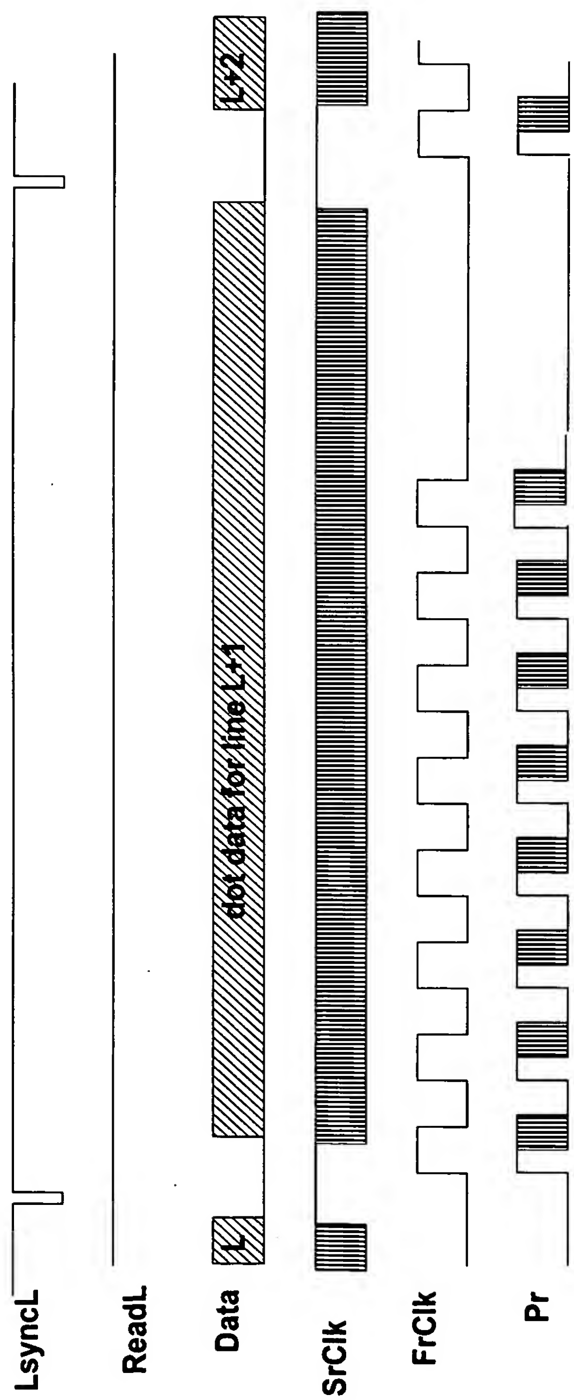


FIG. 317

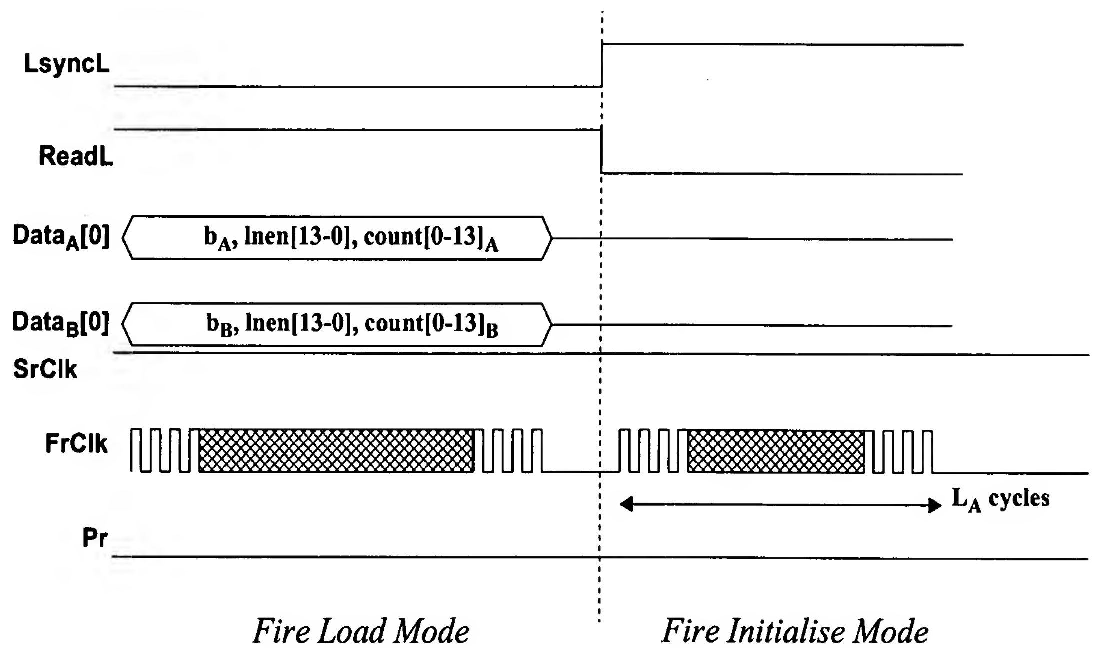

LsyncL
ReadL
DataA[0]
bA, lnen[13-0], count[0-13]A
DataB[0]
bB, lnen[13-0], count[0-13]B
SrClk
FrClk
LA cycles
Pr
Fire Load Mode
Fire Initialise Mode

FIG. 318

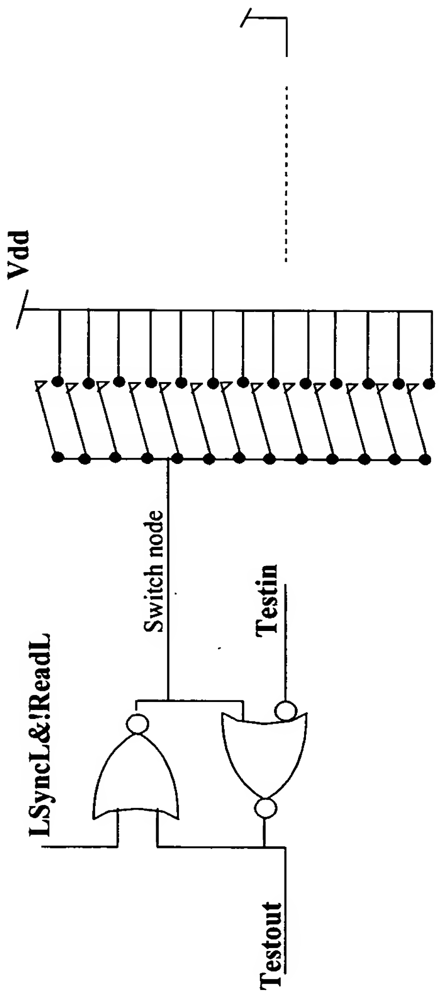


FIG. 319

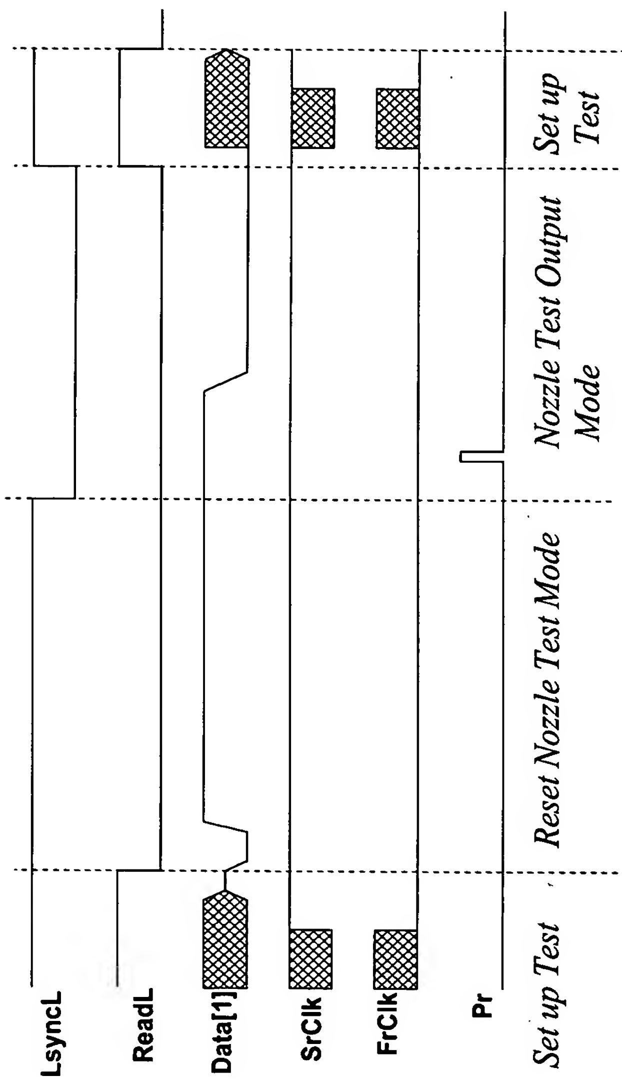

LsyncL
ReadL
Data[1]
SrClk
FrClk
Pr
Set up Test
Reset Nozzle Test Mode
Nozzle Test Output Mode
Set up Test

FIG. 320

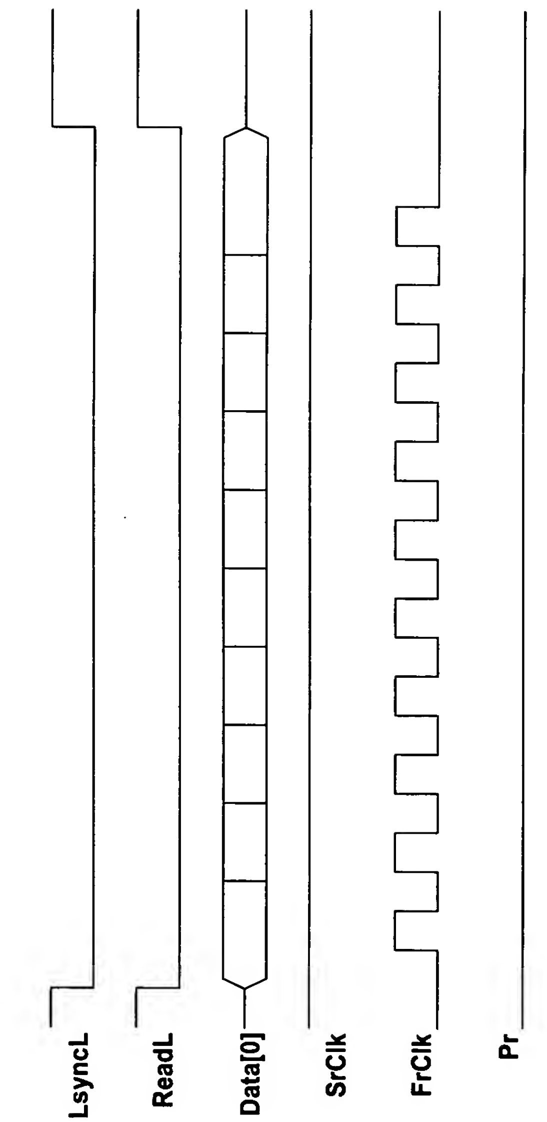


FIG. 321

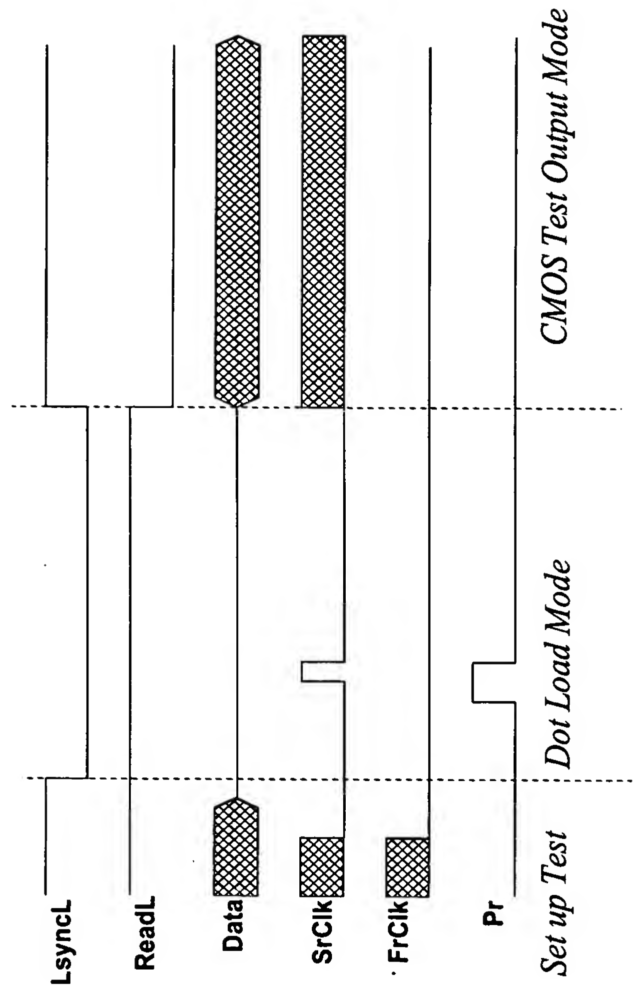

LsyncL
ReadL
Data
SrClk
FrClk
Pr
Set up Test
Dot Load Mode
CMOS Test Output Mode

FIG. 322

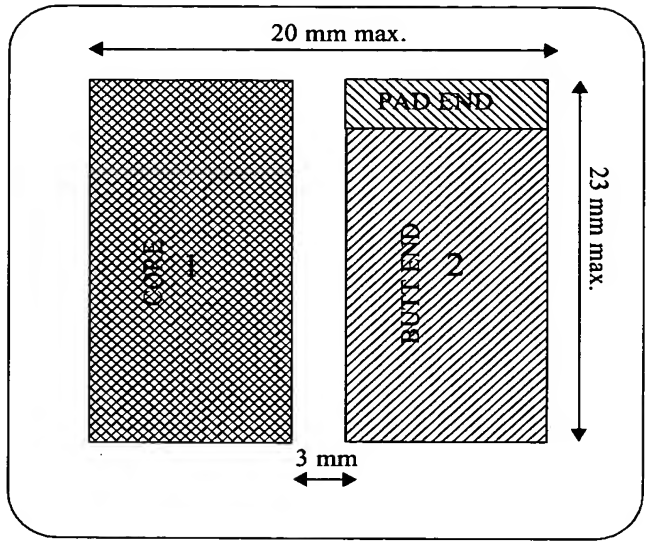


FIG. 323

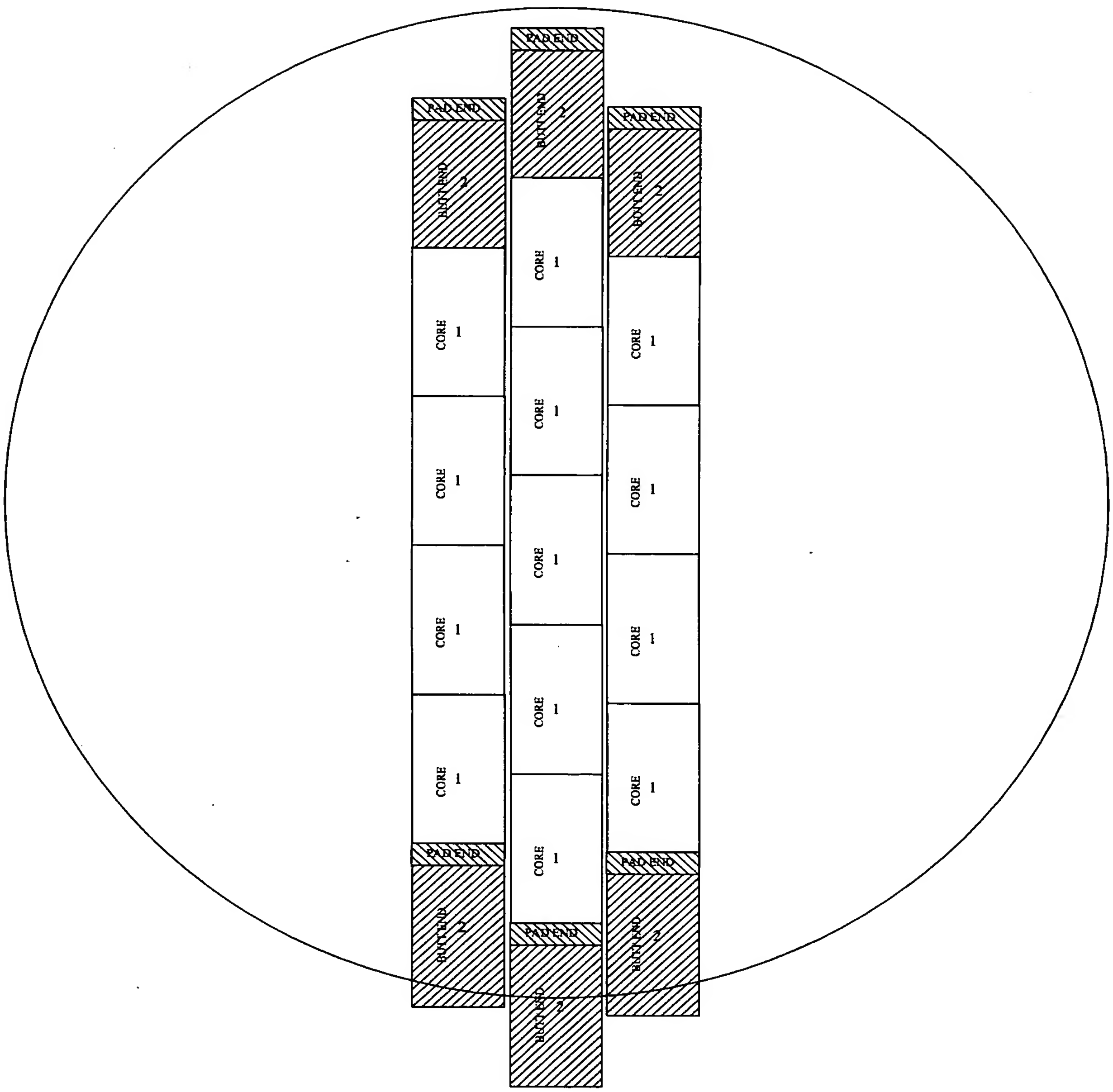


FIG. 324

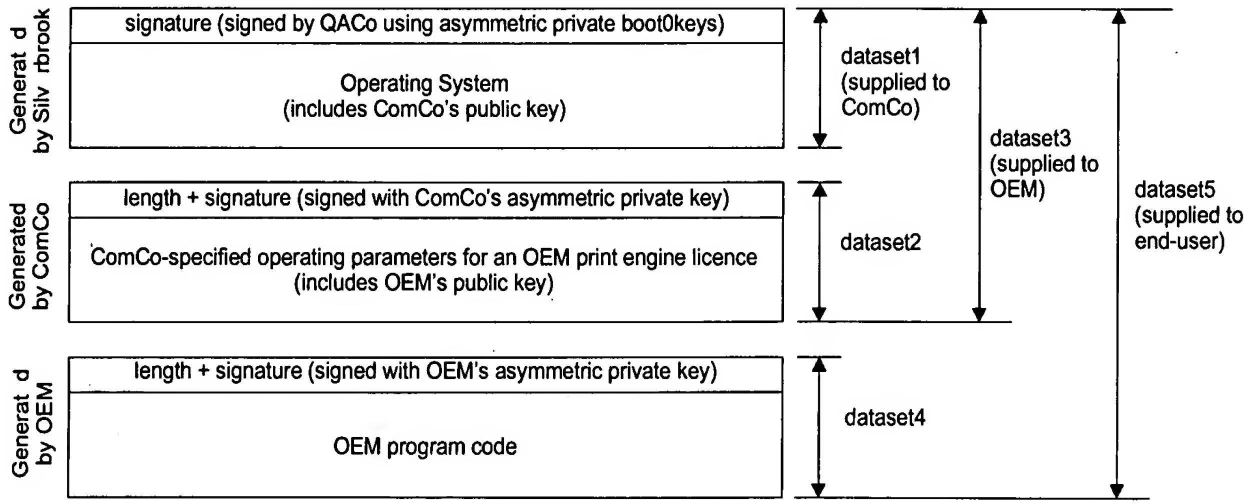


FIG. 325

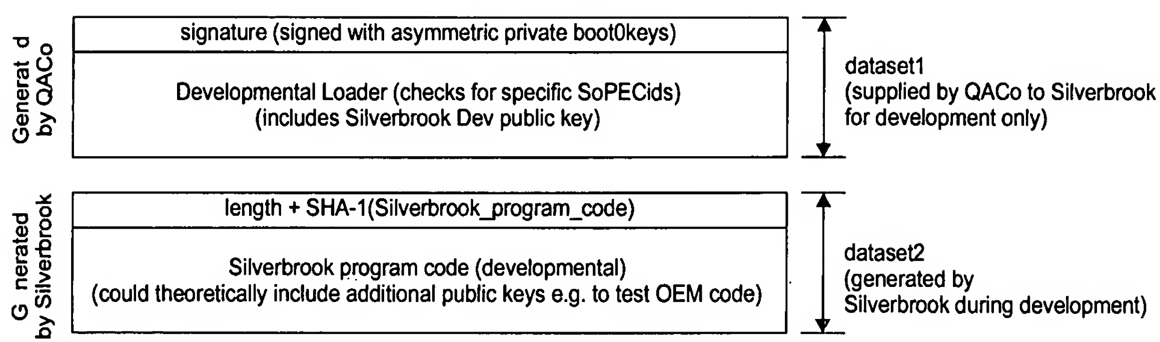


FIG. 327

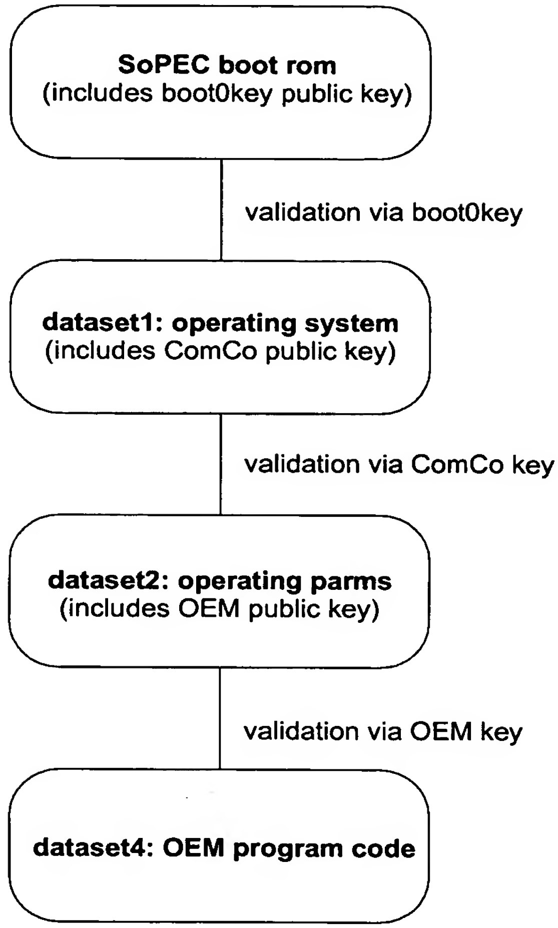

SoPEC boot rom
(includes boot0key public key)
validation via boot0key
dataset1: operating system
(includes ComCo public key)
validation via ComCo key
dataset2: operating parms
(includes OEM public key)
validation via OEM key
dataset4: OEM program code

FIG. 326

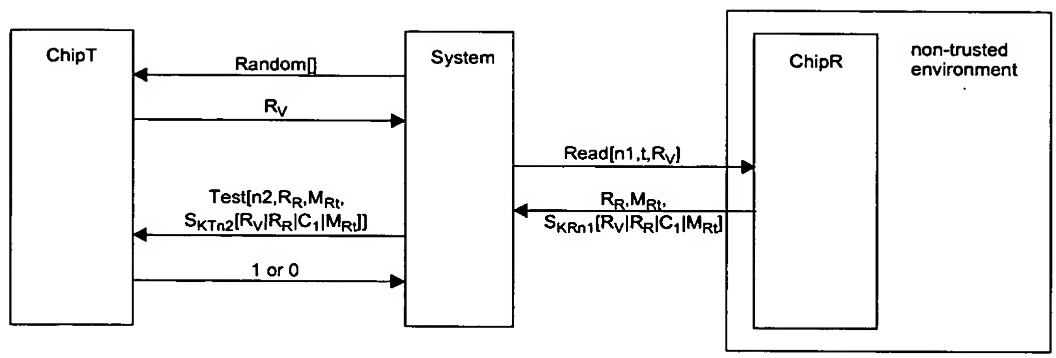


FIG. 328

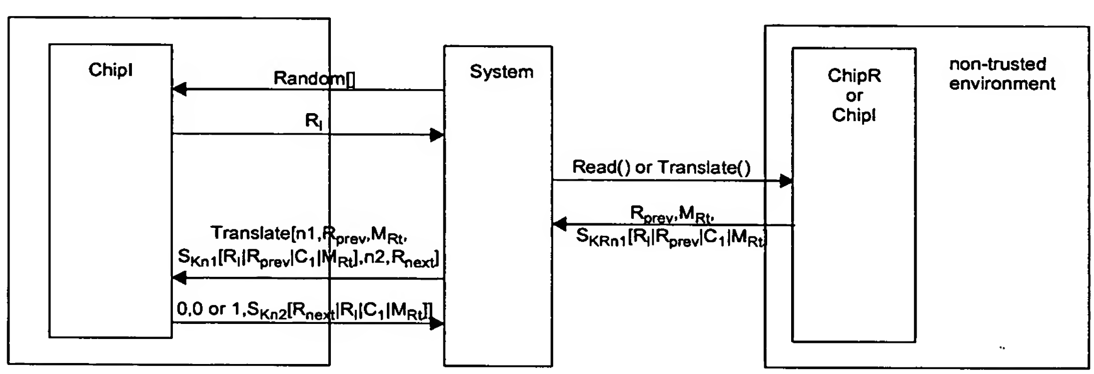


FIG. 329

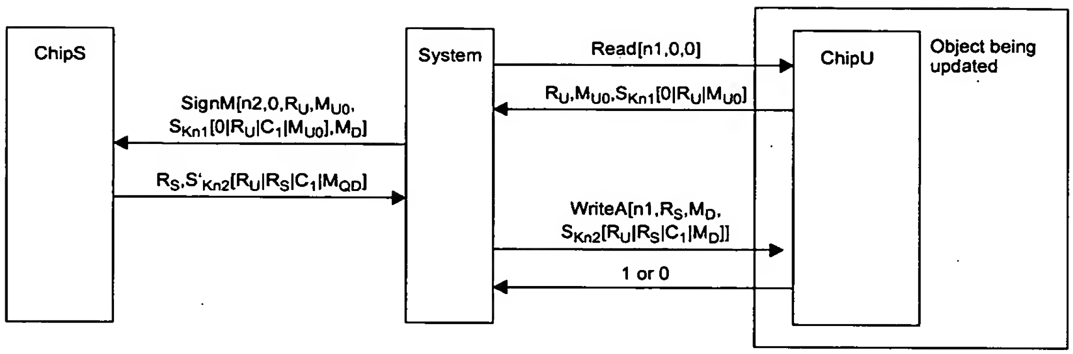


FIG. 330

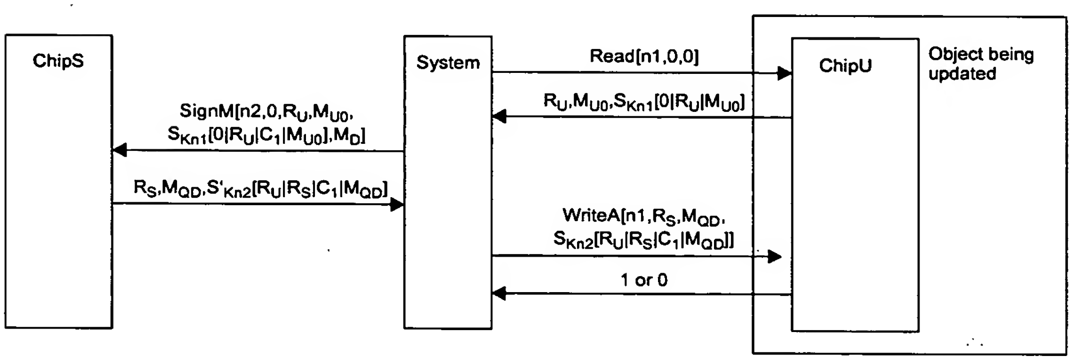


FIG. 331

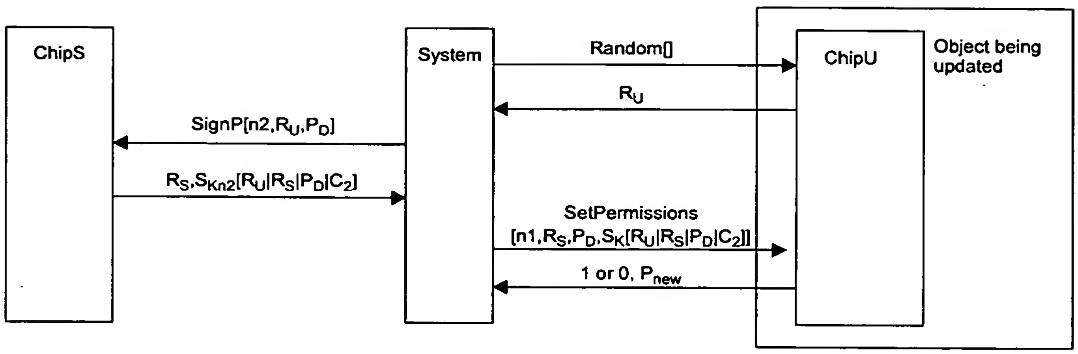


FIG. 332

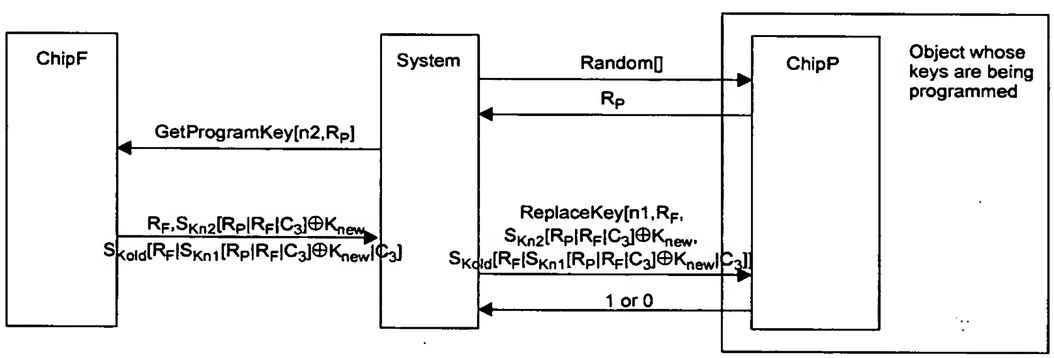


FIG. 333

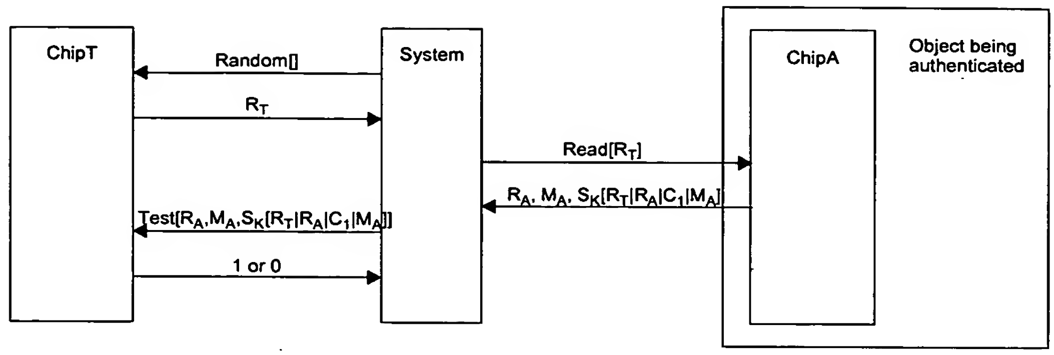


FIG. 334

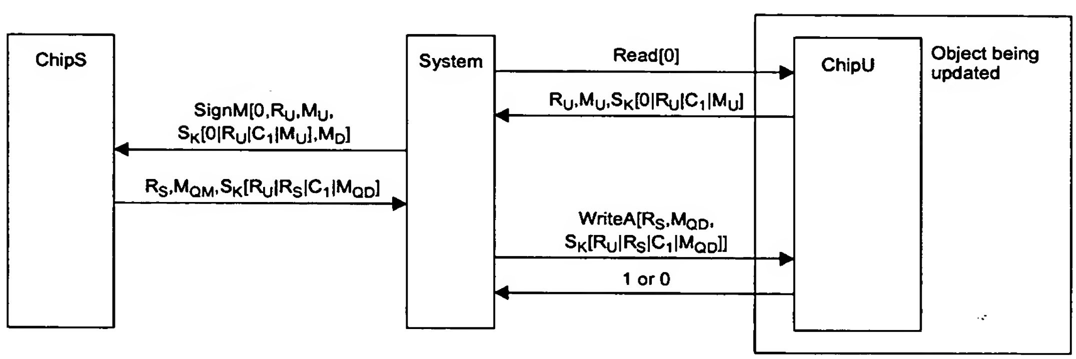


FIG. 335

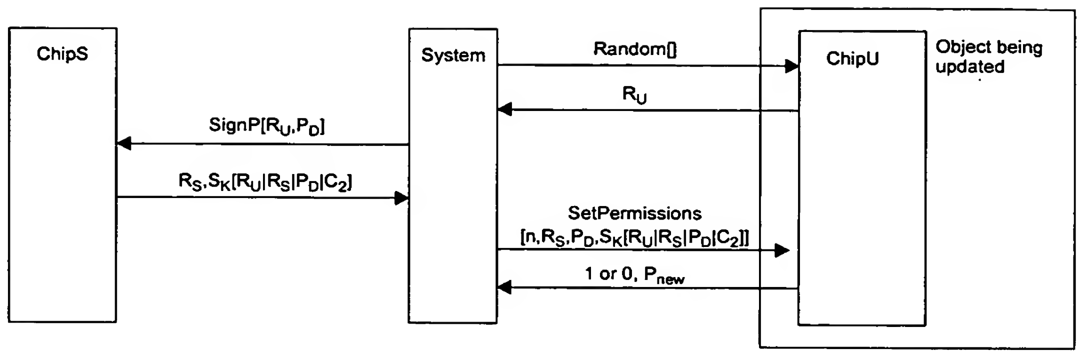


FIG. 336

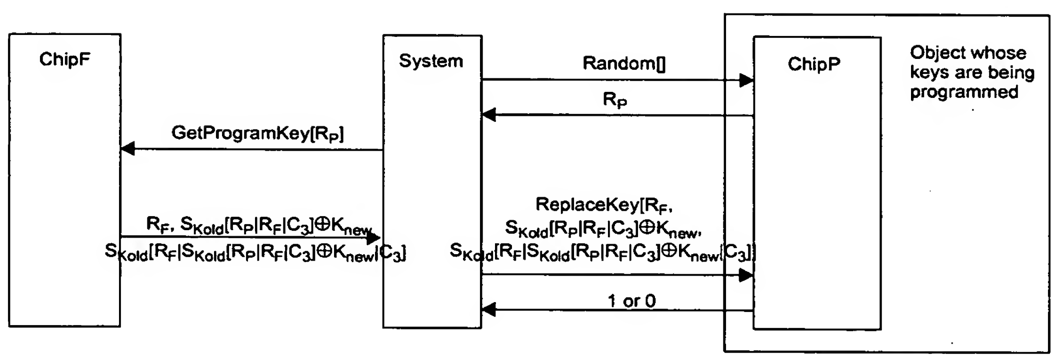


FIG. 337

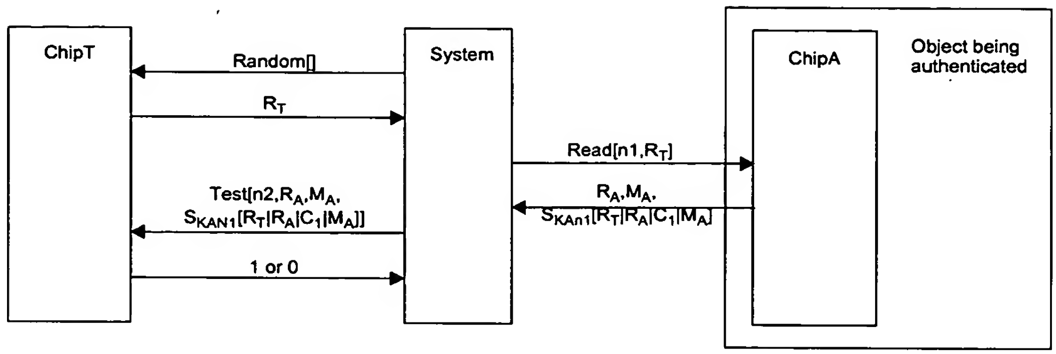


FIG. 338

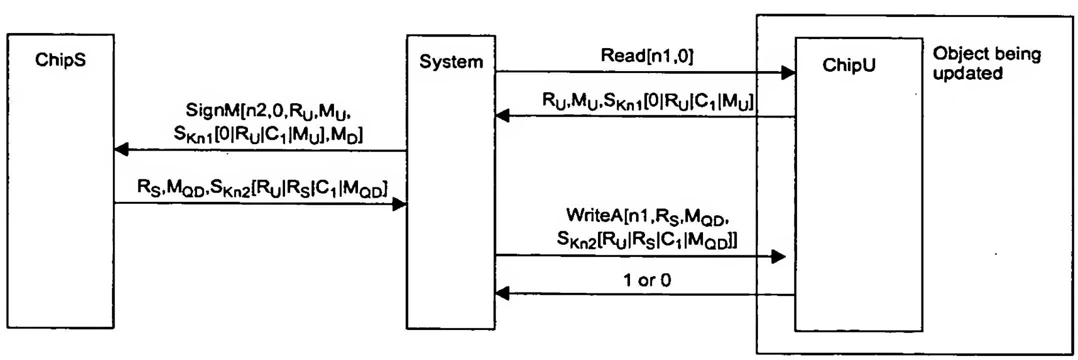


FIG. 339

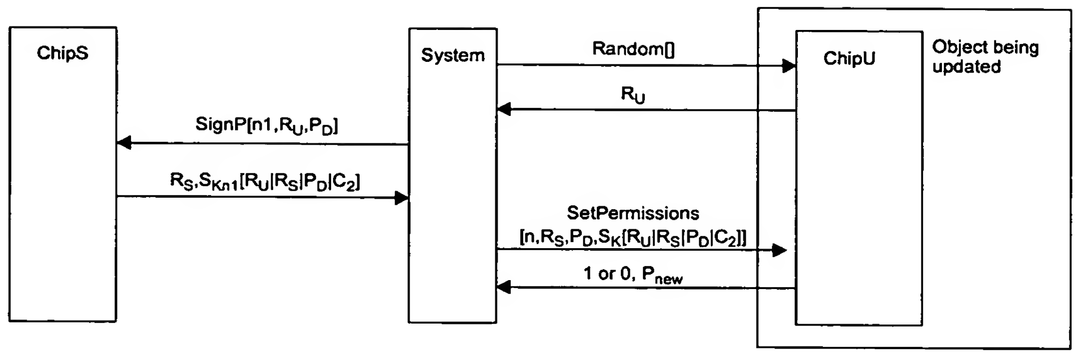


FIG. 340

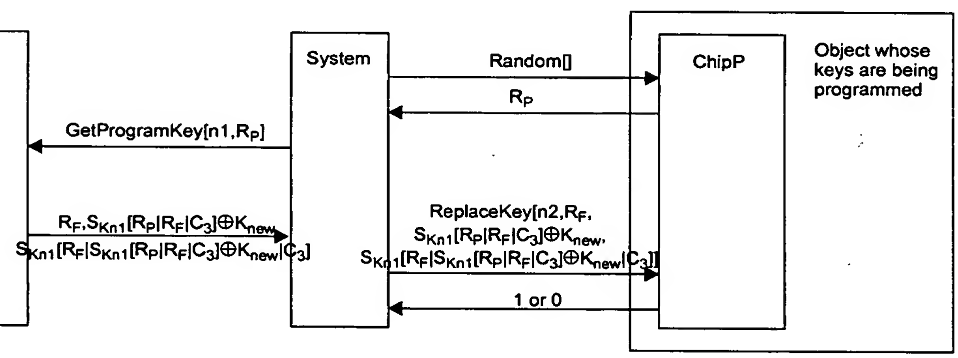


FIG. 341

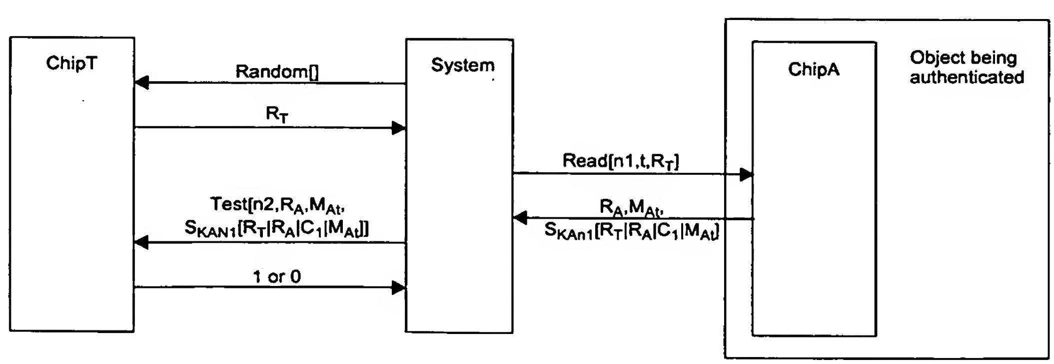


FIG. 342

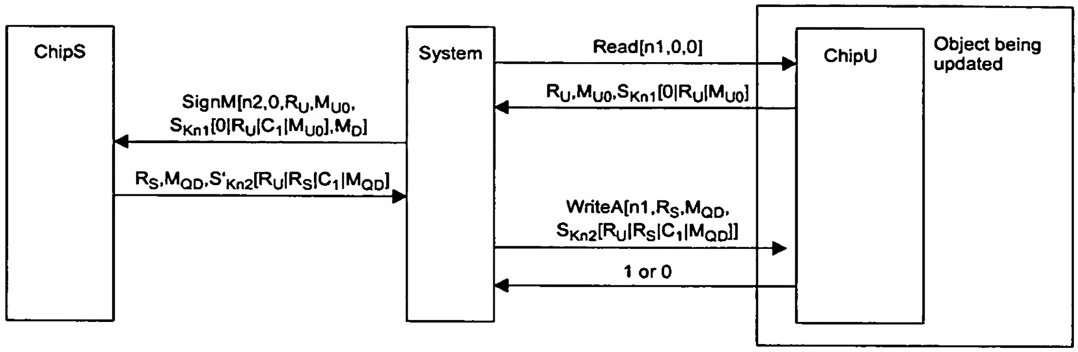


FIG. 343

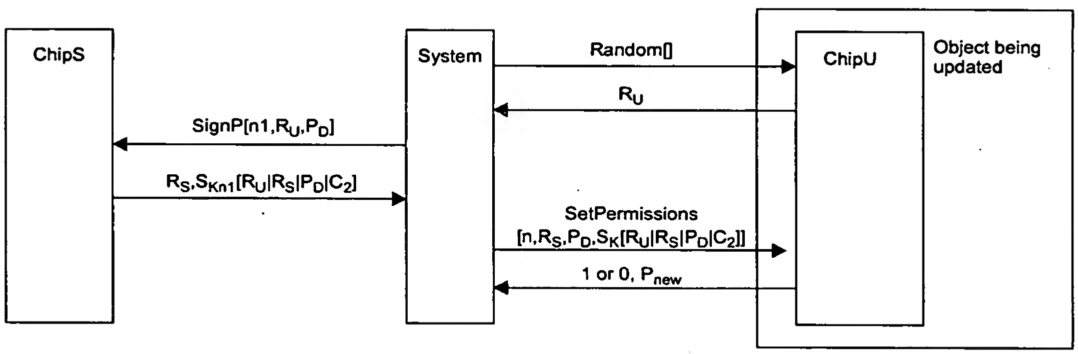


FIG. 344

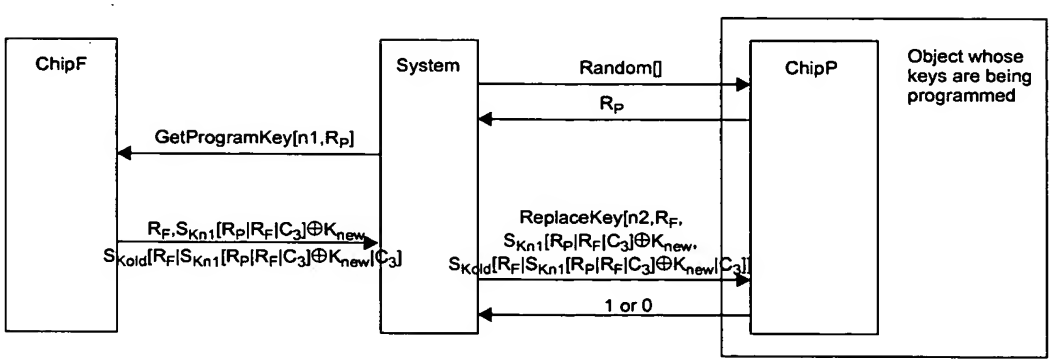


FIG. 345

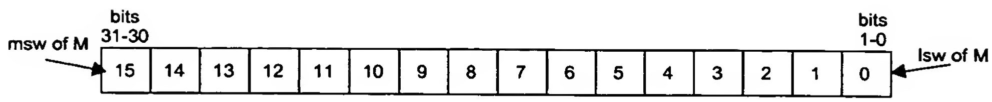


FIG. 346

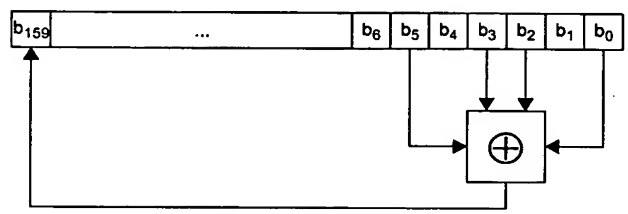


FIG. 347

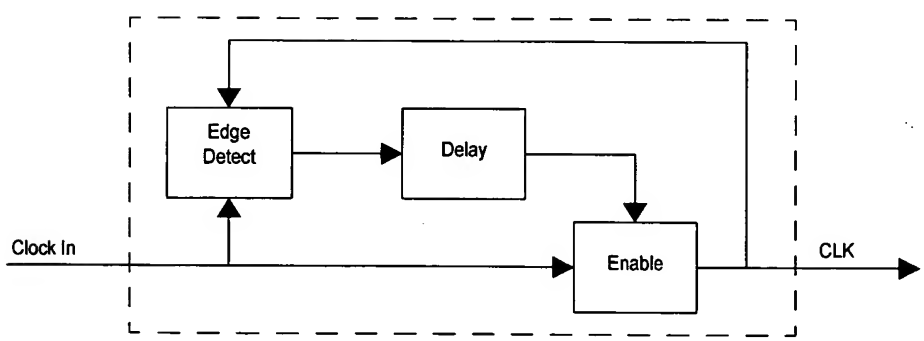


FIG. 348

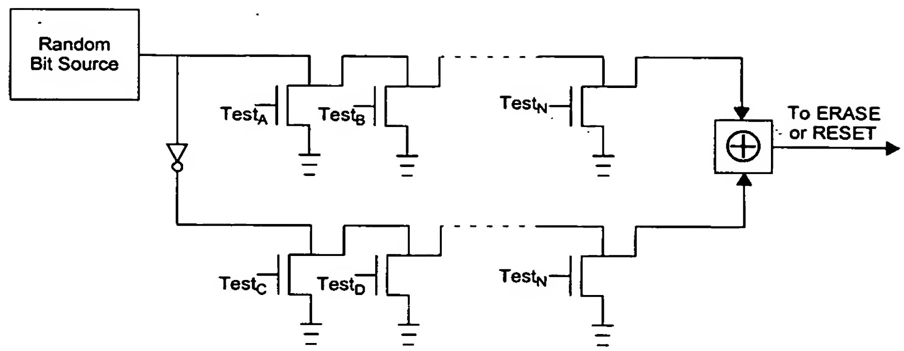


FIG. 349

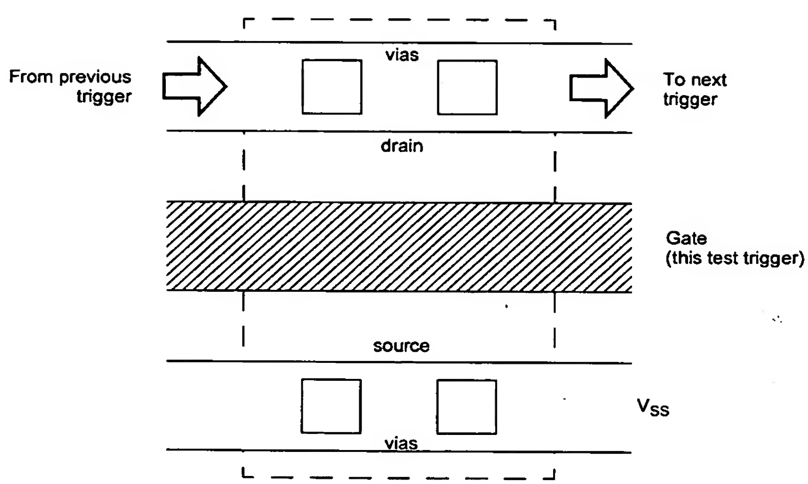


FIG. 350

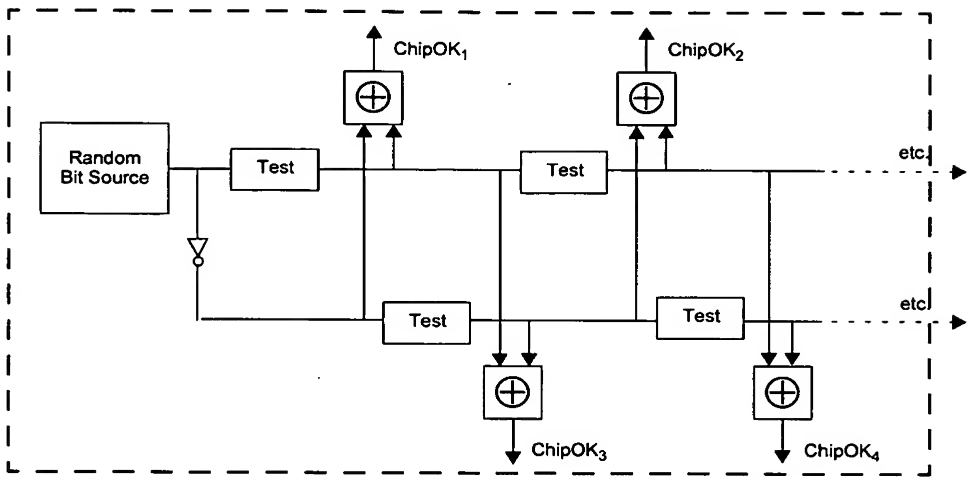


FIG. 351

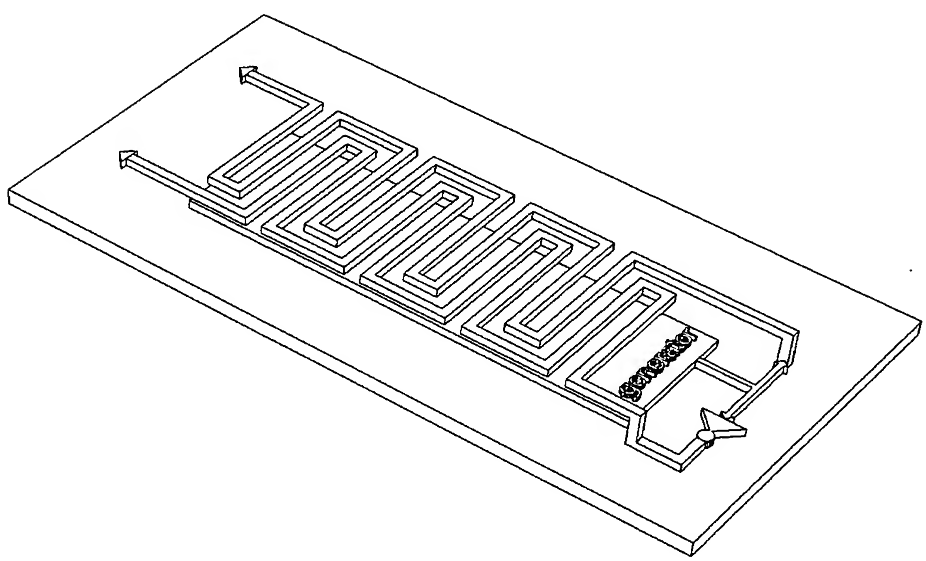


FIG. 352

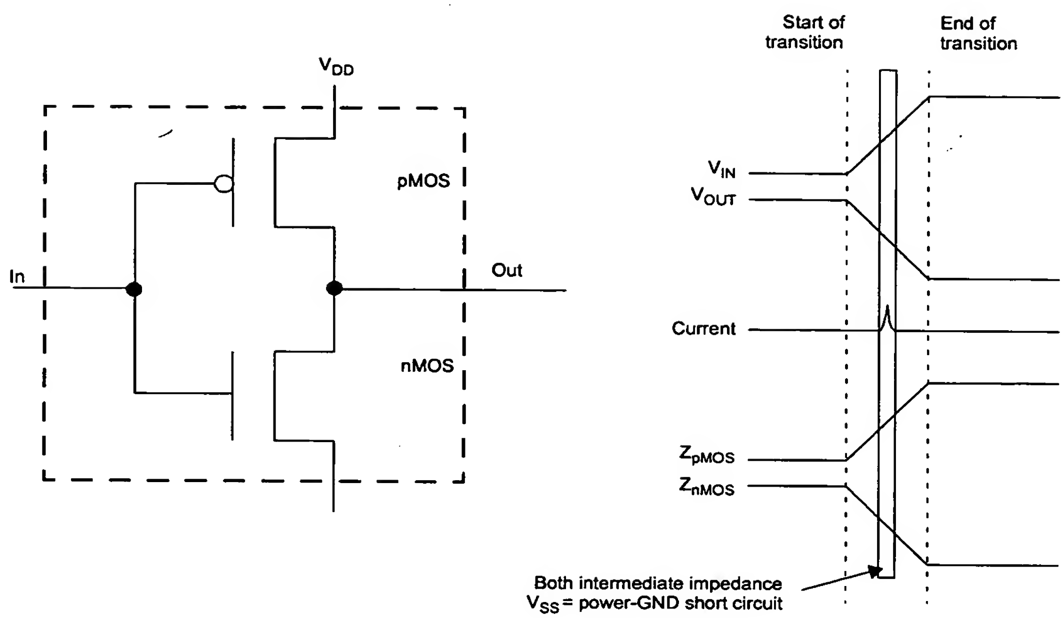


FIG. 353

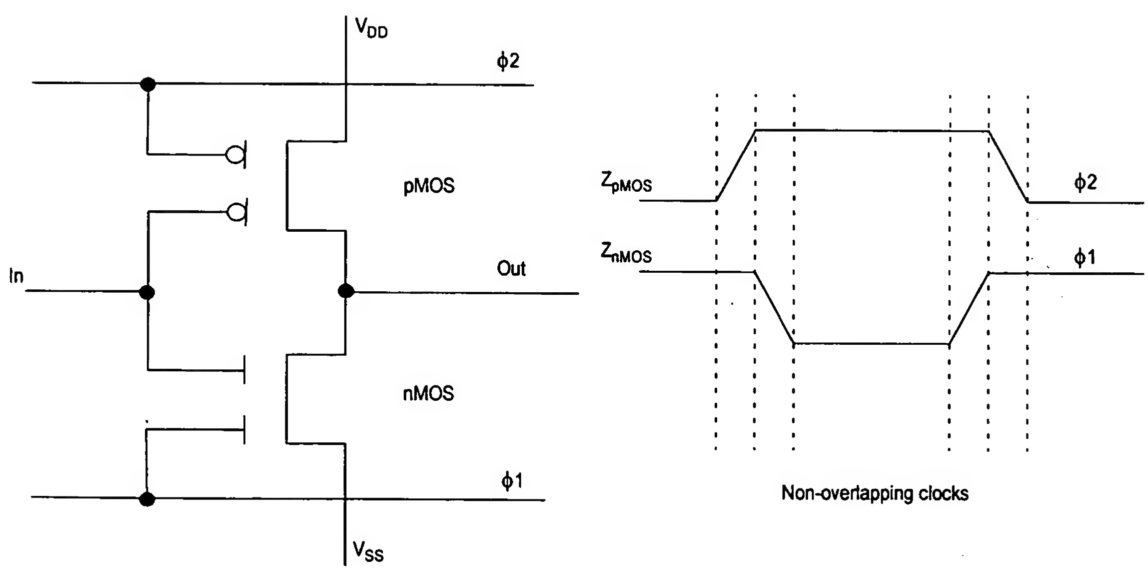


FIG. 354

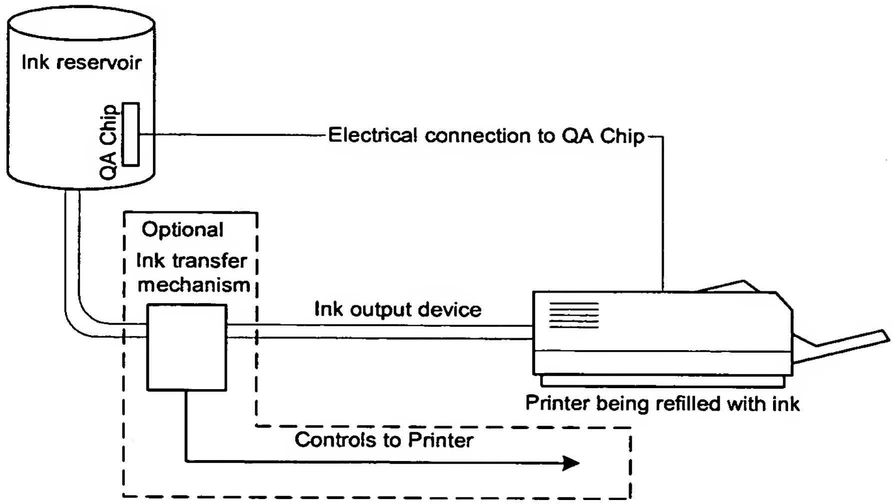


FIG. 355

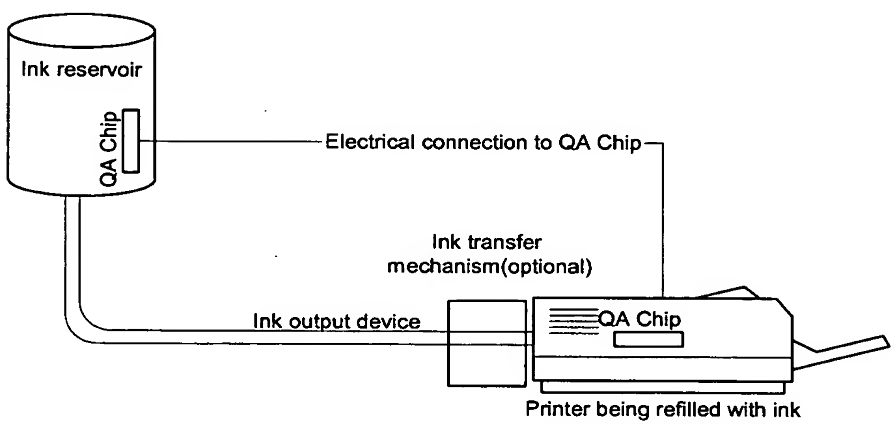


FIG. 356

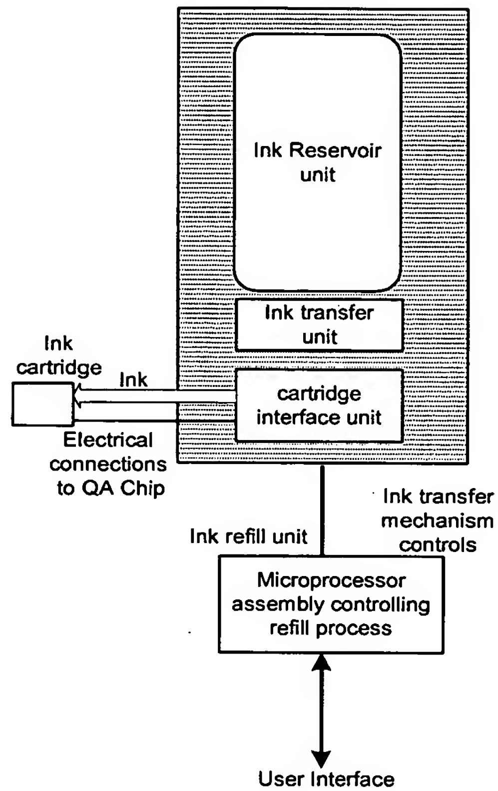


FIG. 357

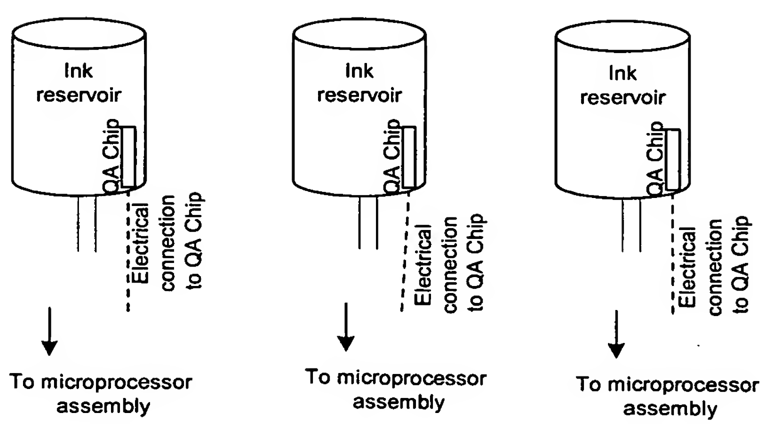


FIG. 358

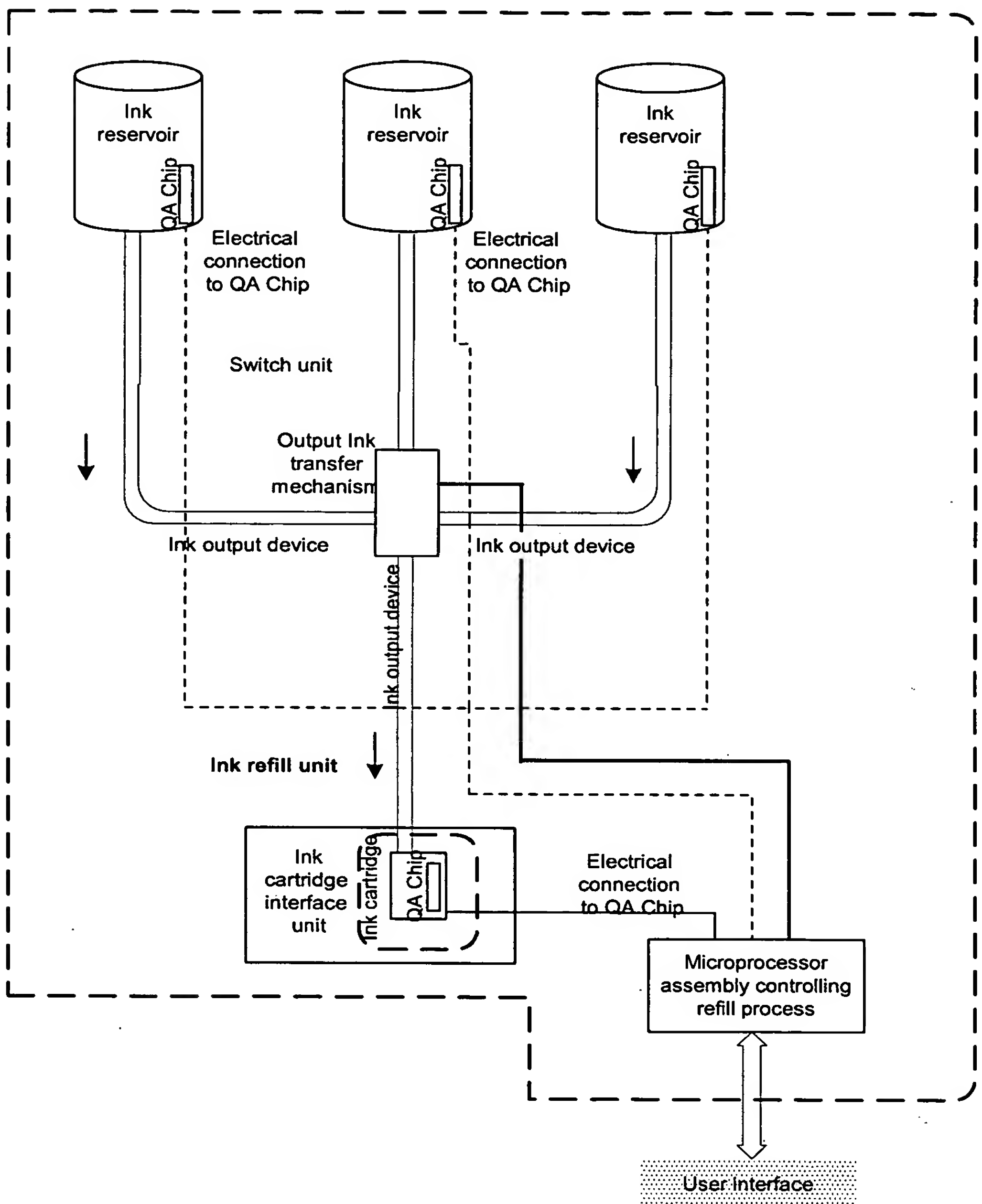

Ink
reservoir
QA Chip
Ink
reservoir
QA Chip
Ink
reservoir
QA Chip
Electrical
connection
to QA Chip
Electrical
connection
to QA Chip
Switch unit
Output Ink
transfer
mechanism
Ink output device
Ink output device
Ink output device
Ink refill unit
Ink
cartridge
interface
unit
Ink cartridge
QA Chip
Electrical
connection
to QA Chip
Microprocessor
assembly controlling
refill process
User Interface

FIG. 359

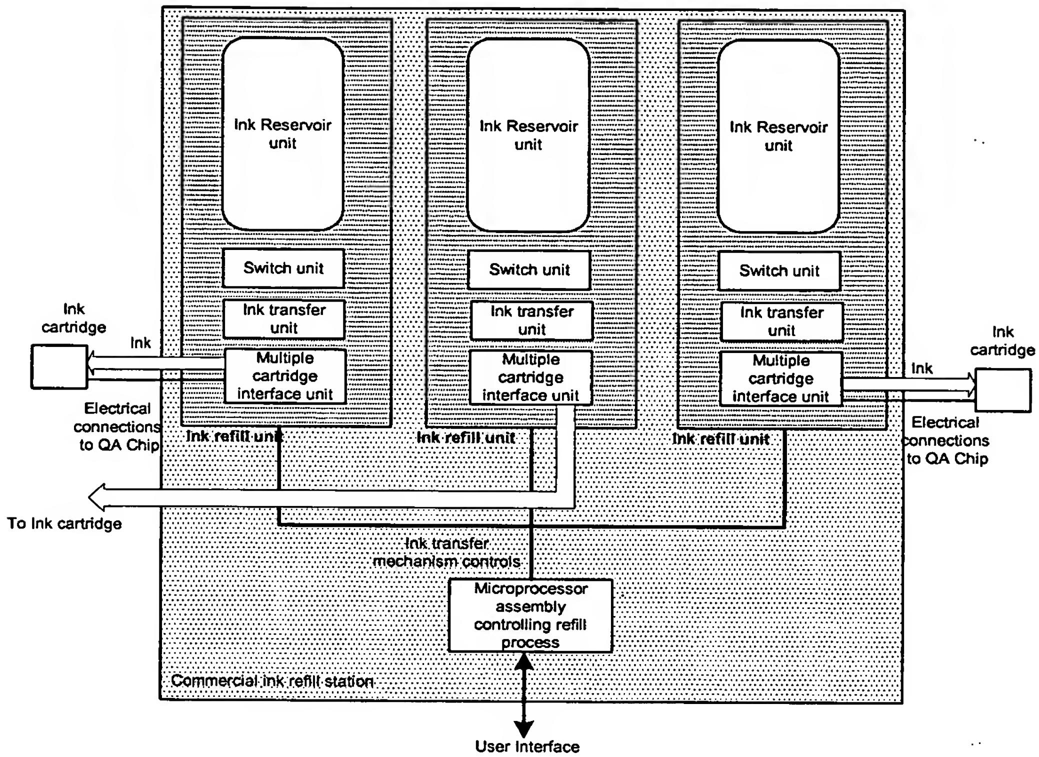


FIG. 360

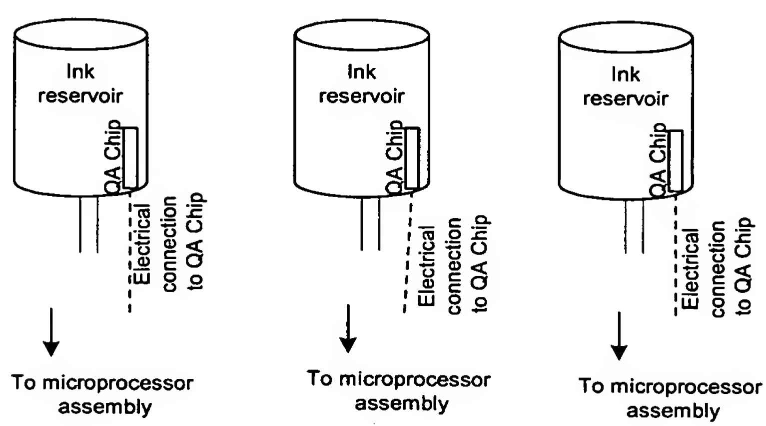


FIG. 361

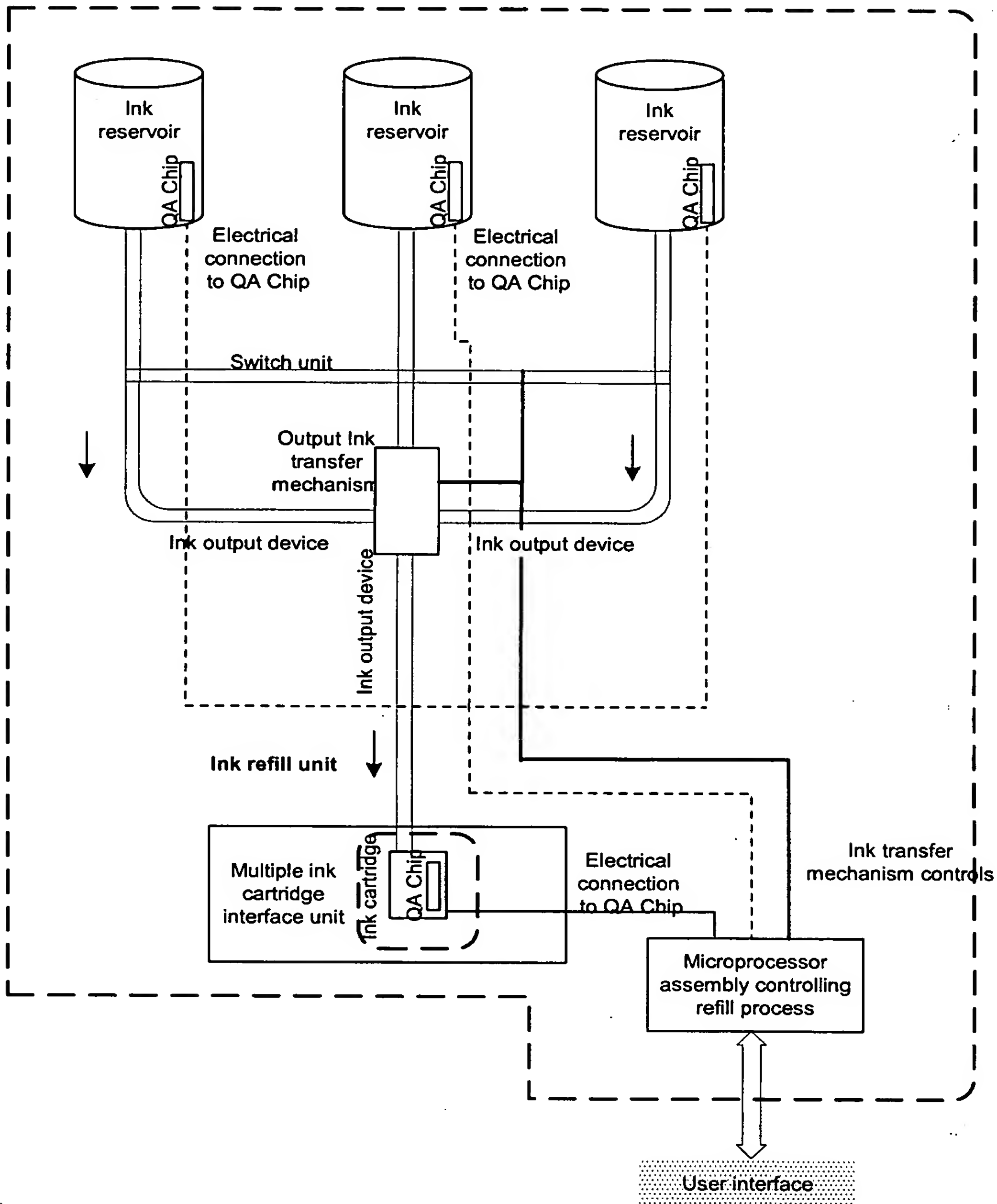

Ink reservoir
QA Chip
Ink reservoir
QA Chip
Ink reservoir
QA Chip
Electrical connection to QA Chip
Electrical connection to QA Chip
Switch unit
Output Ink transfer mechanism
Ink output device
Ink output device
Ink output device
Ink refill unit
Multiple ink cartridge interface unit
Ink cartridge
QA Chip
Electrical connection to QA Chip
Ink transfer mechanism controls
Microprocessor assembly controlling refill process
User interface

FIG. 362

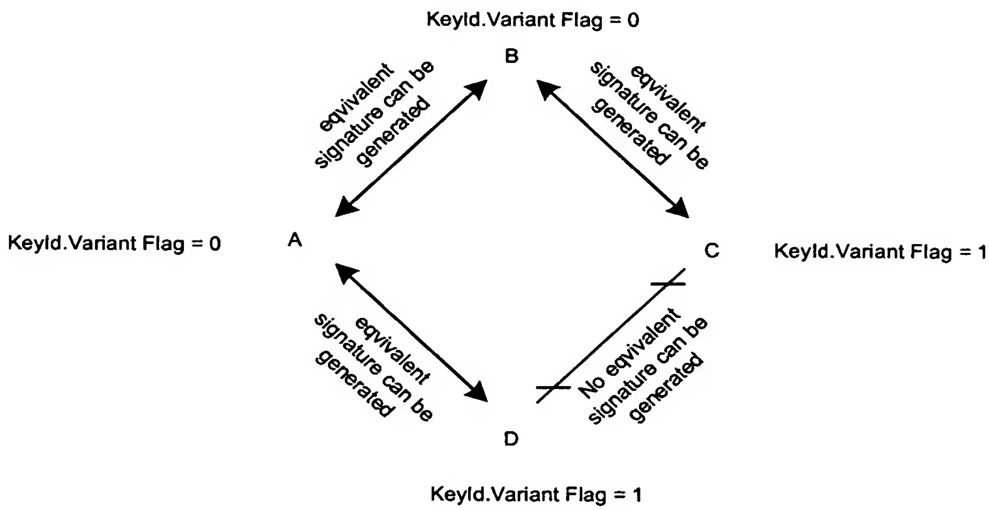


FIG. 363

31–17	16–4	3–0
Type (15 bits)	Permissions (13 bits)	Size and Position (4 bits)

FIG. 364

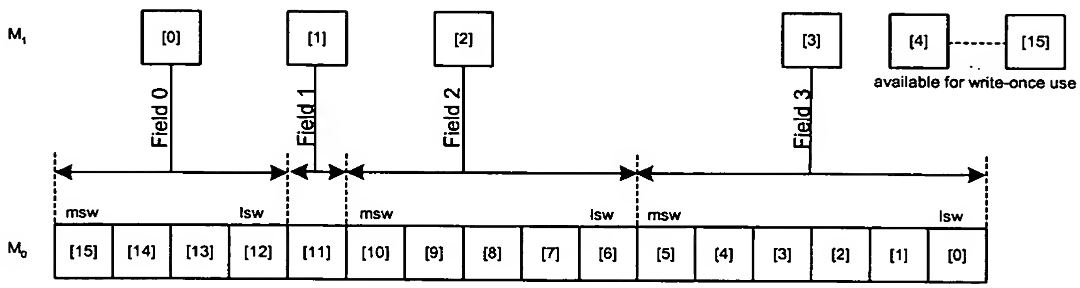


FIG. 365

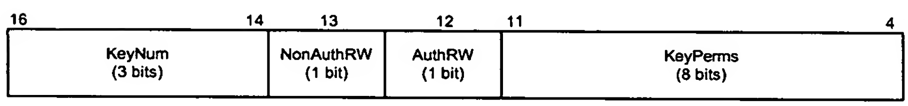


FIG. 366

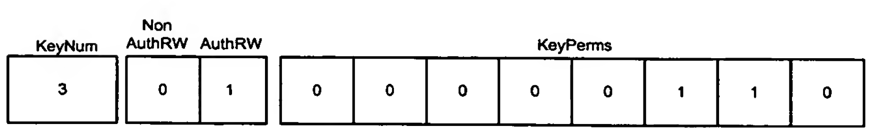


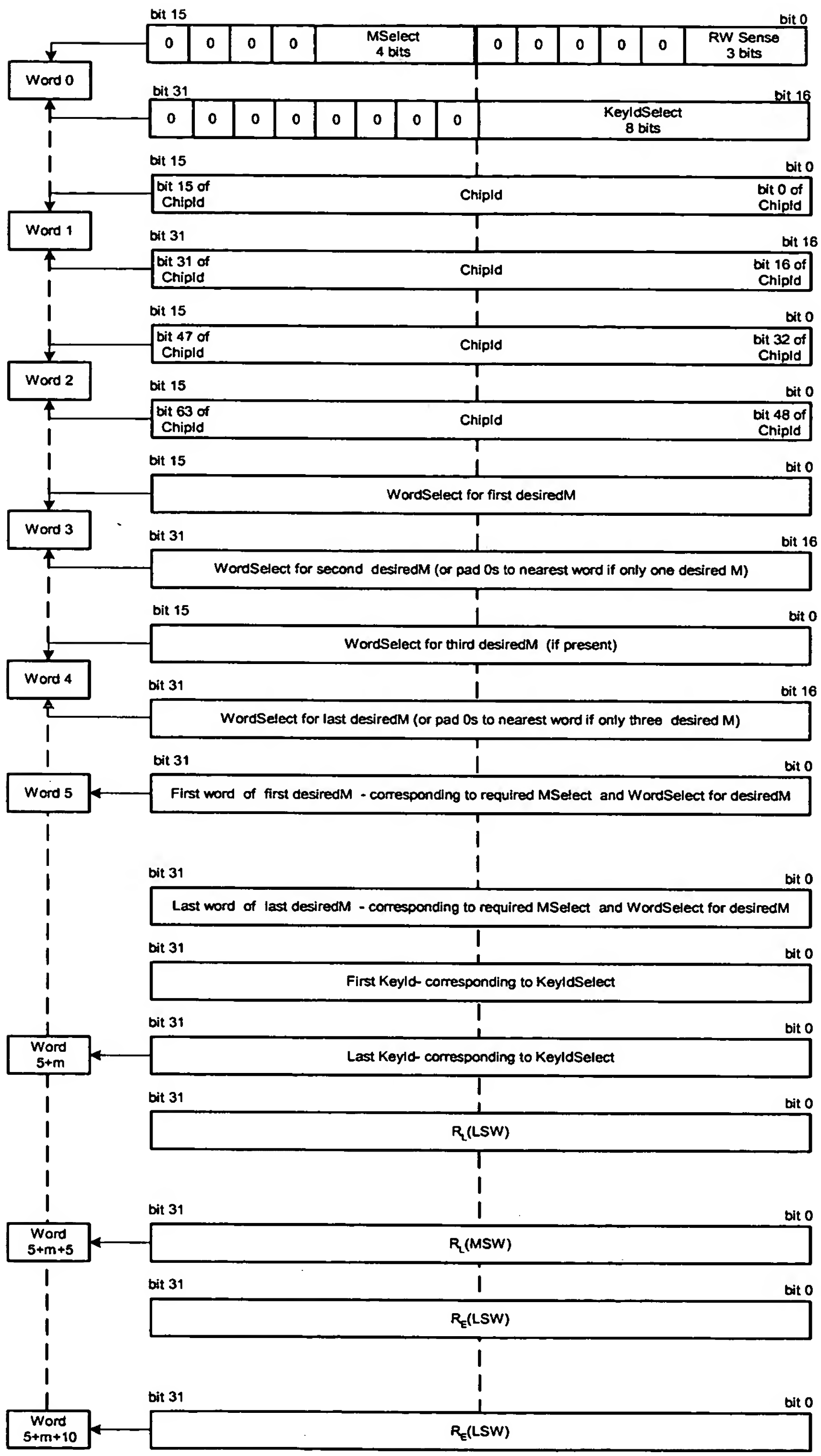
FIG. 367

KeyNum	Non AuthRW	AuthRW	KeyPerms							
3	1	1	0	0	0	0	1	1	1	0

FIG. 368

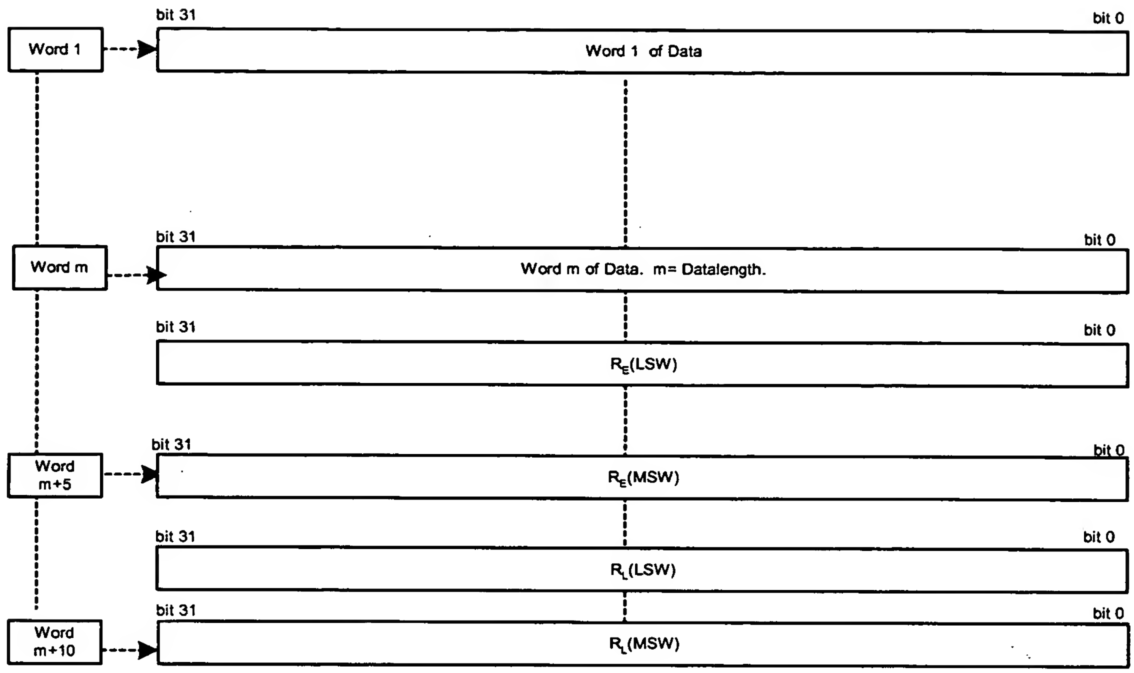
31 17	16 14	13	12	11 4	3 0
Type (15 bits)	KeyNum (3 bits)	NonAuth RW (1 bit)	AuthRW (1 bit)	KeyPerms (8 bits)	EndPos (4 bits)

FIG. 369



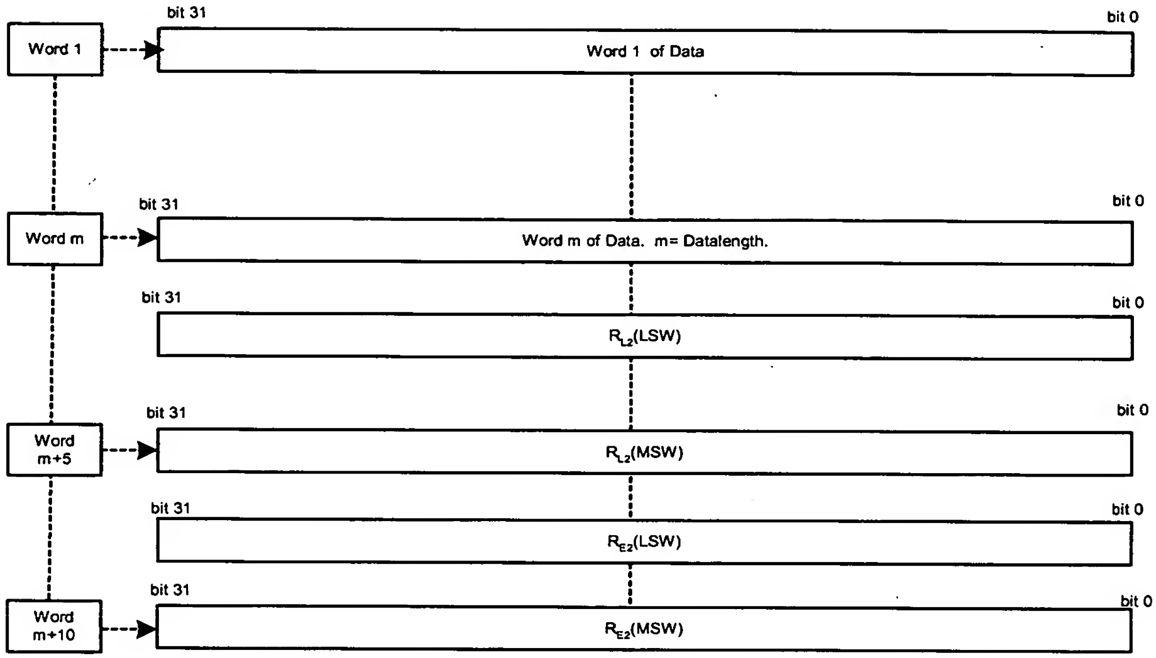
bit 15
bit 0
0
0
0
0
MSelect 4 bits
0
0
0
0
0
RW Sense 3 bits
Word 0
bit 31
bit 16
0
0
0
0
0
0
0
0
KeyIdSelect 8 bits
bit 15 of ChipId
ChipId
bit 0 of ChipId
Word 1
bit 31 of ChipId
bit 16 of ChipId
bit 47 of ChipId
bit 32 of ChipId
Word 2
bit 63 of ChipId
bit 48 of ChipId
WordSelect for first desiredM
Word 3
WordSelect for second desiredM (or pad 0s to nearest word if only one desired M)
WordSelect for third desiredM (if present)
Word 4
WordSelect for last desiredM (or pad 0s to nearest word if only three desired M)
Word 5
First word of first desiredM - corresponding to required MSelect and WordSelect for desiredM
Last word of last desiredM - corresponding to required MSelect and WordSelect for desiredM
First KeyId- corresponding to KeyIdSelect
Word 5+m
Last KeyId- corresponding to KeyIdSelect
R_L(LSW)
Word 5+m+5
R_L(MSW)
R_E(LSW)
Word 5+m+10
R_E(LSW)

FIG. 370



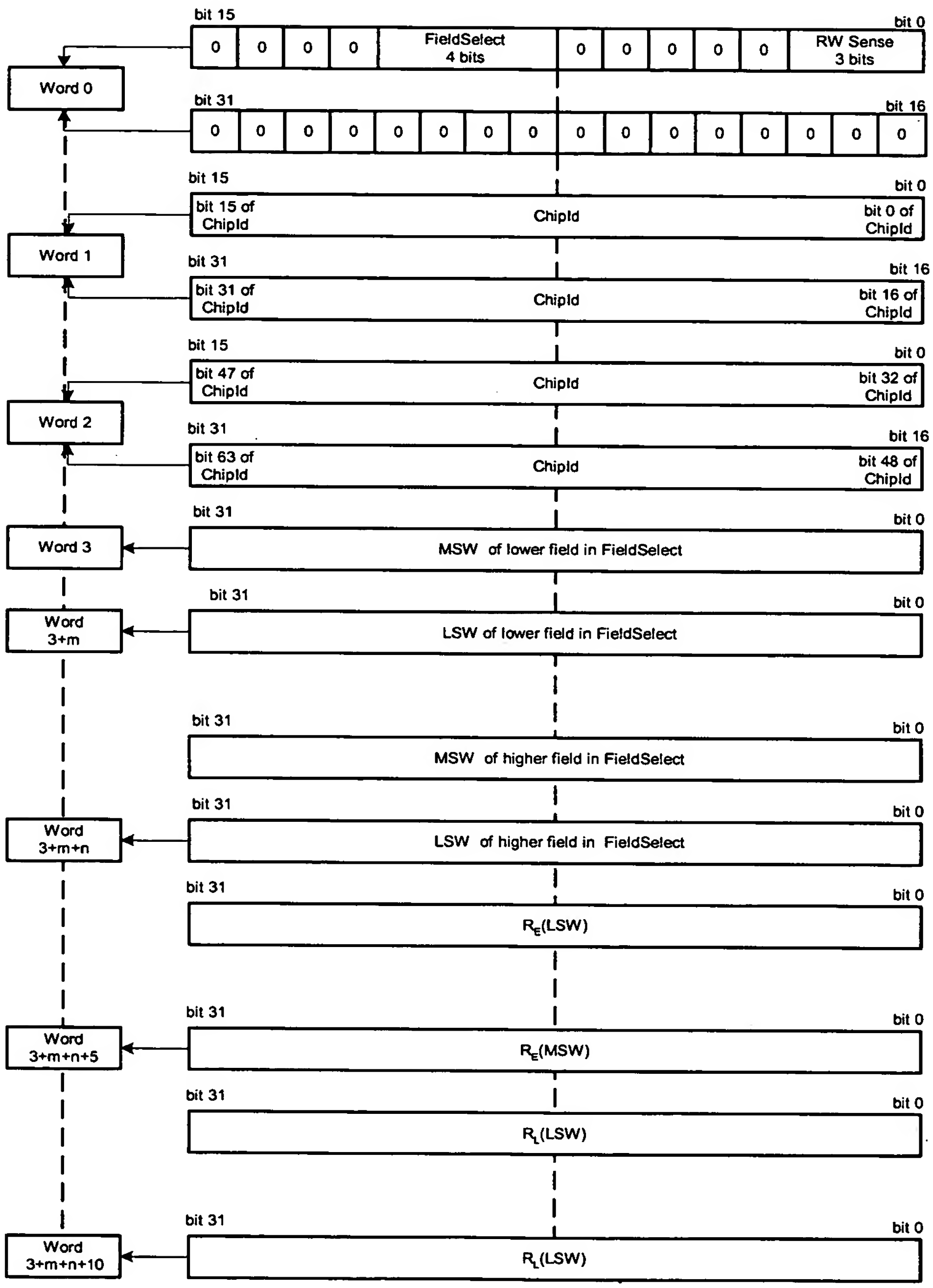
Word 1
bit 31
bit 0
Word 1 of Data
Word m
bit 31
bit 0
Word m of Data. m= Datalength.
bit 31
bit 0
R_E(LSW)
Word m+5
bit 31
bit 0
R_E(MSW)
bit 31
bit 0
R_L(LSW)
Word m+10
bit 31
bit 0
R_L(MSW)

FIG. 371



Word 1
bit 31
bit 0
Word 1 of Data
Word m
bit 31
bit 0
Word m of Data. m= Datalength.
bit 31
bit 0
R_{L2}(LSW)
Word m+5
bit 31
bit 0
R_{L2}(MSW)
bit 31
bit 0
R_{E2}(LSW)
Word m+10
bit 31
bit 0
R_{E2}(MSW)

FIG. 372



bit 15
bit 0
0
0
0
0
FieldSelect 4 bits
0
0
0
0
0
RW Sense 3 bits
Word 0
bit 31
bit 16
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
bit 15
bit 0
bit 15 of ChipId
ChipId
bit 0 of ChipId
Word 1
bit 31
bit 16
bit 31 of ChipId
ChipId
bit 16 of ChipId
bit 15
bit 0
bit 47 of ChipId
ChipId
bit 32 of ChipId
Word 2
bit 31
bit 16
bit 63 of ChipId
ChipId
bit 48 of ChipId
bit 31
bit 0
Word 3
MSW of lower field in FieldSelect
bit 31
bit 0
Word 3+m
LSW of lower field in FieldSelect
bit 31
bit 0
MSW of higher field in FieldSelect
bit 31
bit 0
Word 3+m+n
LSW of higher field in FieldSelect
bit 31
bit 0
R_E(LSW)
bit 31
bit 0
Word 3+m+n+5
R_E(MSW)
bit 31
bit 0
R_L(LSW)
bit 31
bit 0
Word 3+m+n+10
R_L(LSW)

FIG. 373

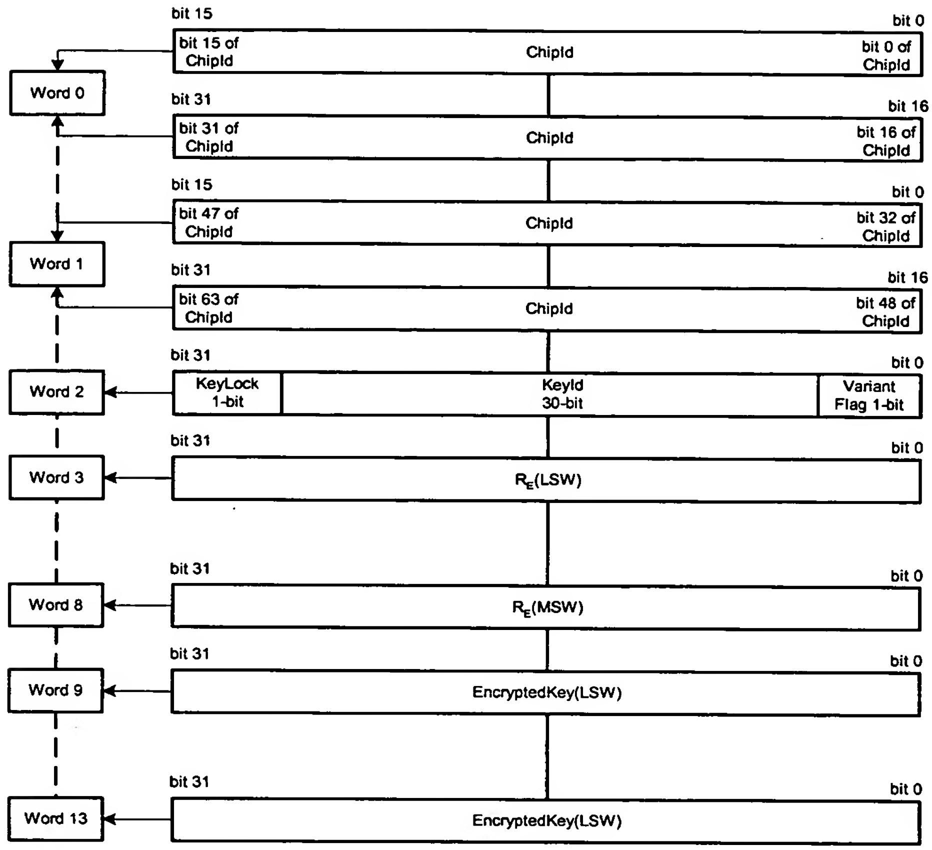

bit 15
bit 0
bit 15 of ChipId
ChipId
bit 0 of ChipId
Word 0
bit 31
bit 16
bit 31 of ChipId
ChipId
bit 16 of ChipId
bit 15
bit 0
bit 47 of ChipId
ChipId
bit 32 of ChipId
Word 1
bit 31
bit 16
bit 63 of ChipId
ChipId
bit 48 of ChipId
bit 31
bit 0
Word 2
KeyLock 1-bit
KeyId 30-bit
Variant Flag 1-bit
bit 31
bit 0
Word 3
R_E(LSW)
bit 31
bit 0
Word 8
R_E(MSW)
bit 31
bit 0
Word 9
EncryptedKey(LSW)
bit 31
bit 0
Word 13
EncryptedKey(LSW)

FIG. 374

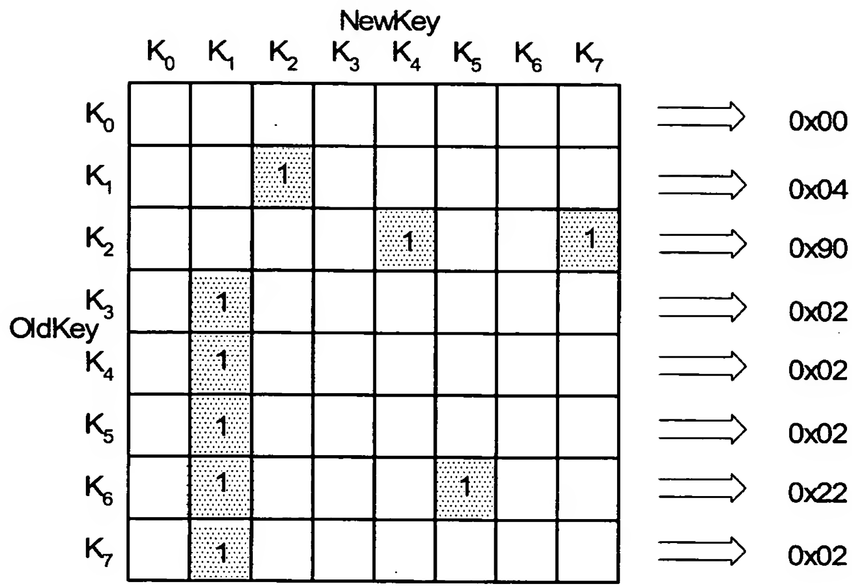


FIG. 375

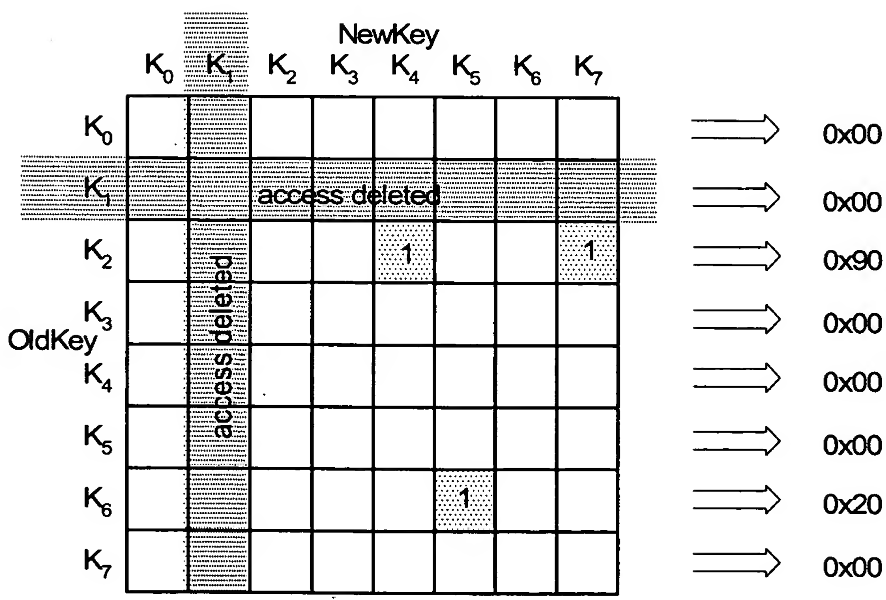


FIG. 376

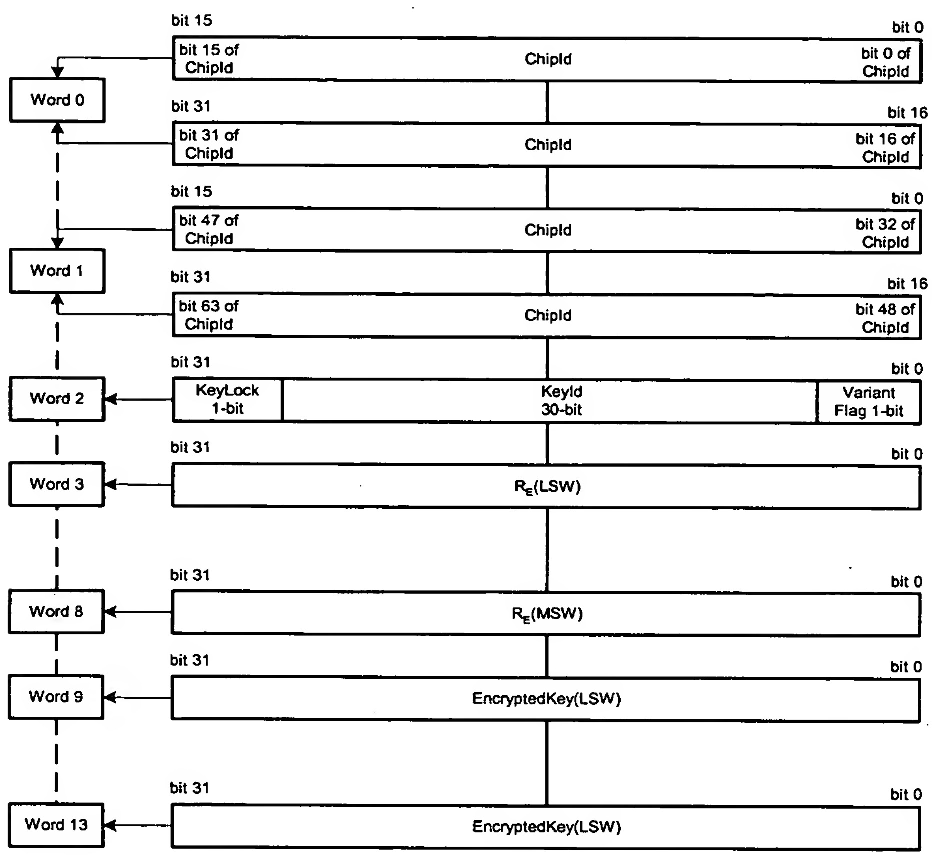

bit 15
bit 0
bit 15 of ChipId
ChipId
bit 0 of ChipId
Word 0
bit 31
bit 16
bit 31 of ChipId
ChipId
bit 16 of ChipId
bit 15
bit 0
bit 47 of ChipId
ChipId
bit 32 of ChipId
Word 1
bit 31
bit 16
bit 63 of ChipId
ChipId
bit 48 of ChipId
bit 31
bit 0
Word 2
KeyLock 1-bit
KeyId 30-bit
Variant Flag 1-bit
bit 31
bit 0
Word 3
R_E(LSW)
bit 31
bit 0
Word 8
R_E(MSW)
bit 31
bit 0
Word 9
EncryptedKey(LSW)
bit 31
bit 0
Word 13
EncryptedKey(LSW)

FIG. 377

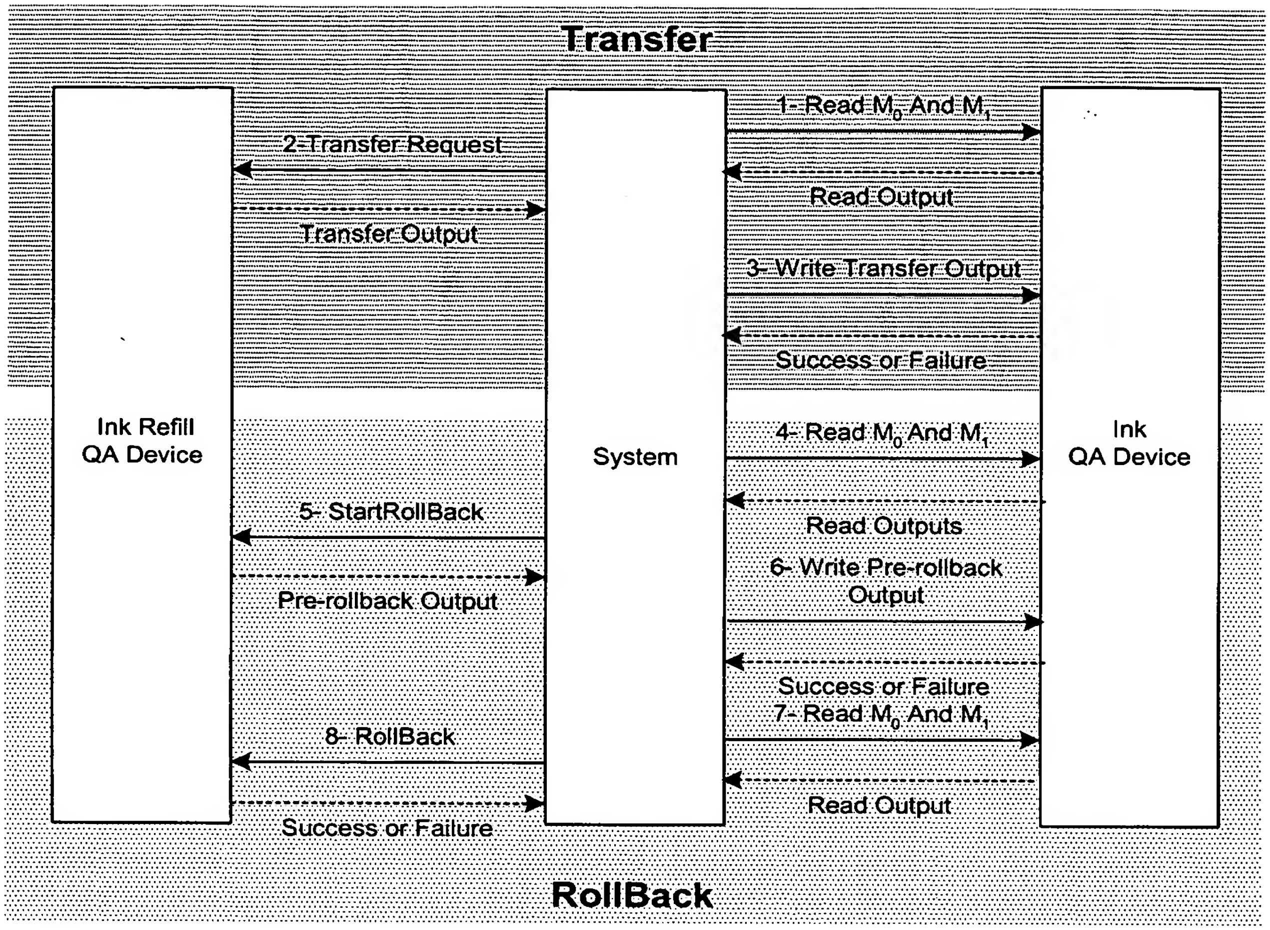
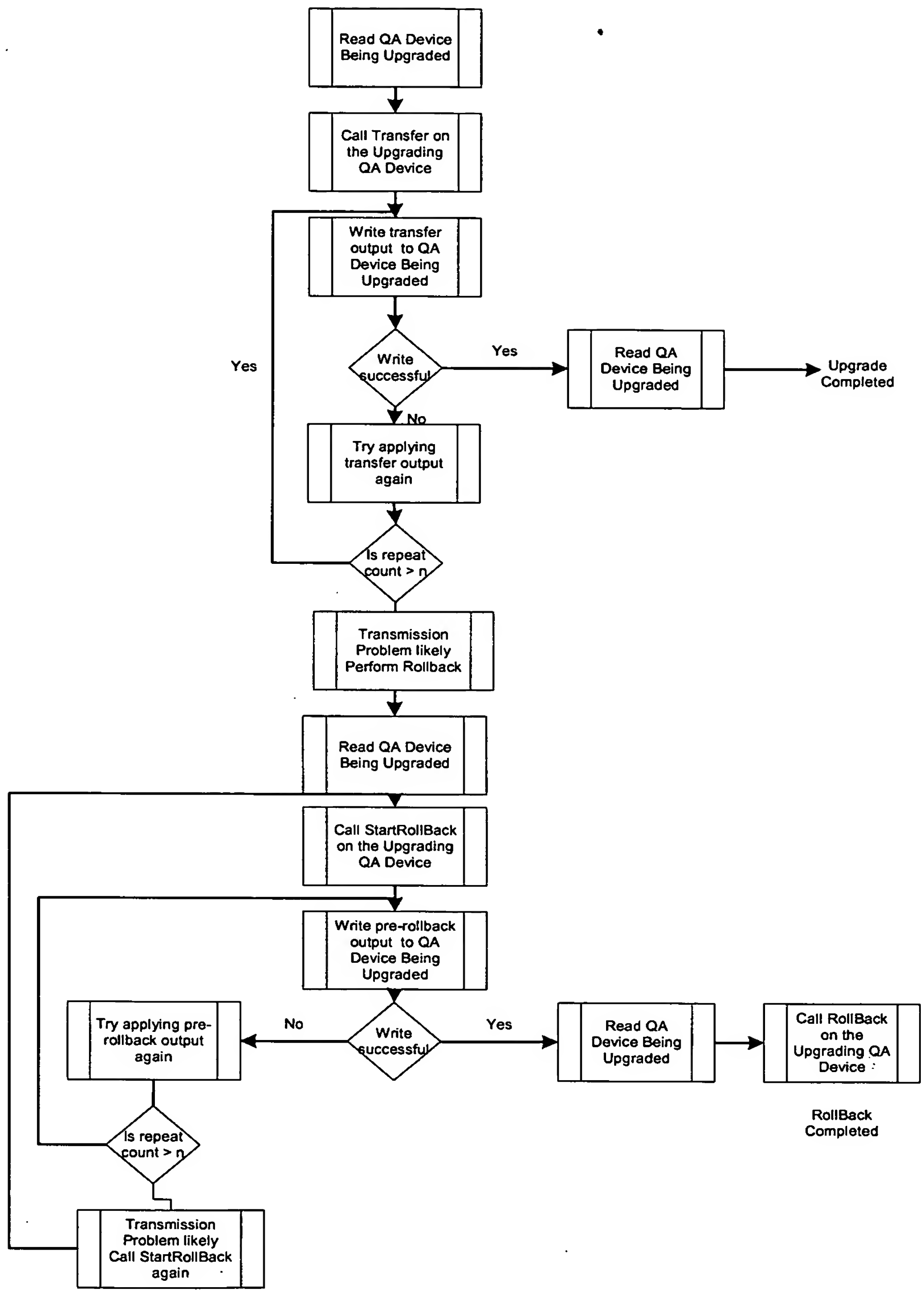

Transfer
1- Read M_0 And M_1
2-Transfer Request
Read Output
Transfer Output
3- Write Transfer Output
Success or Failure
Ink Refill
QA Device
System
4- Read M_0 And M_1
Ink
QA Device
5- StartRollBack
Read Outputs
6- Write Pre-rollback
Output
Pre-rollback Output
Success or Failure
7- Read M_0 And M_1
8- RollBack
Read Output
Success or Failure
RollBack

FIG. 378



Read QA Device Being Upgraded
Call Transfer on the Upgrading QA Device
Write transfer output to QA Device Being Upgraded
Write successful
Yes
Read QA Device Being Upgraded
Upgrade Completed
No
Try applying transfer output again
Is repeat count > n
Yes
Transmission Problem likely Perform Rollback
Read QA Device Being Upgraded
Call StartRollBack on the Upgrading QA Device
Write pre-rollback output to QA Device Being Upgraded
Write successful
Yes
Read QA Device Being Upgraded
Call RollBack on the Upgrading QA Device
RollBack Completed
No
Try applying pre-rollback output again
Is repeat count > n
Transmission Problem likely Call StartRollBack again

FIG. 379

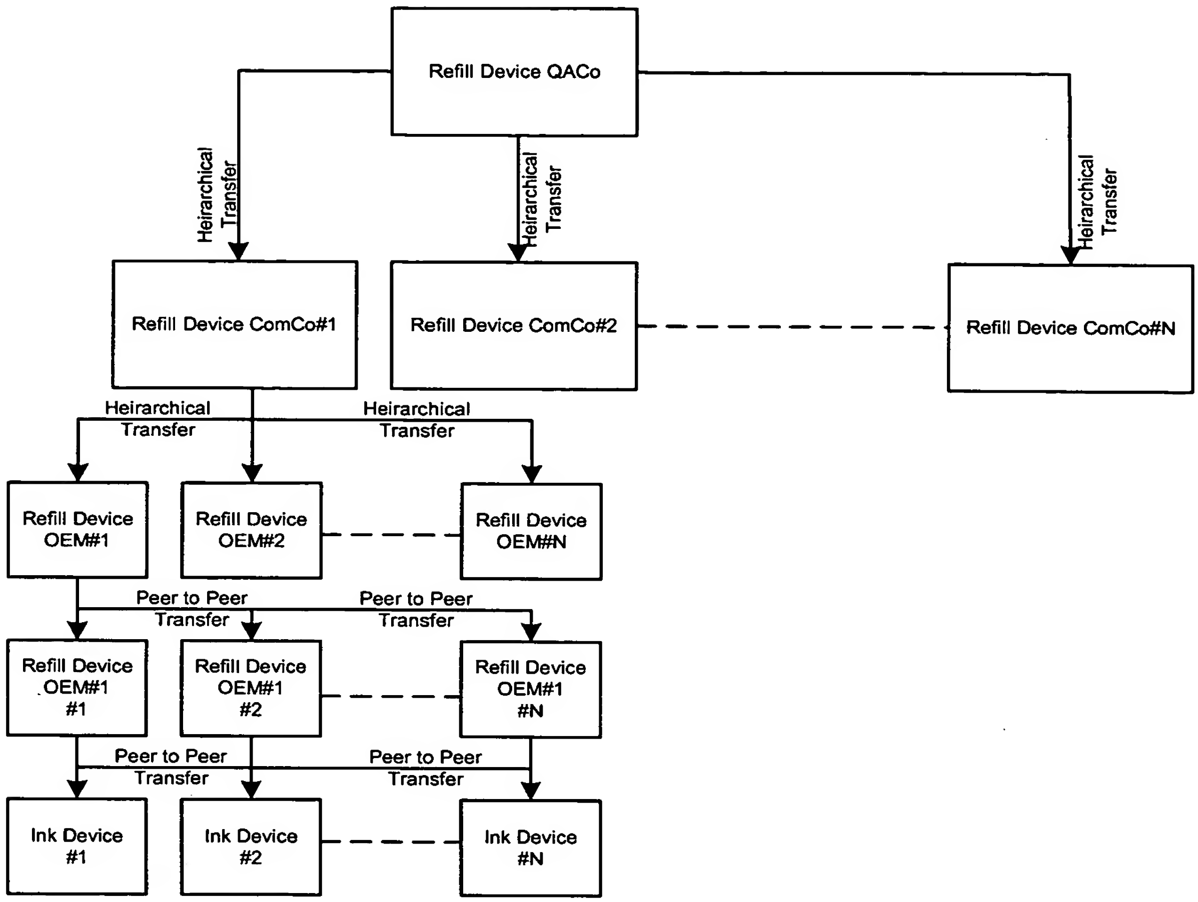
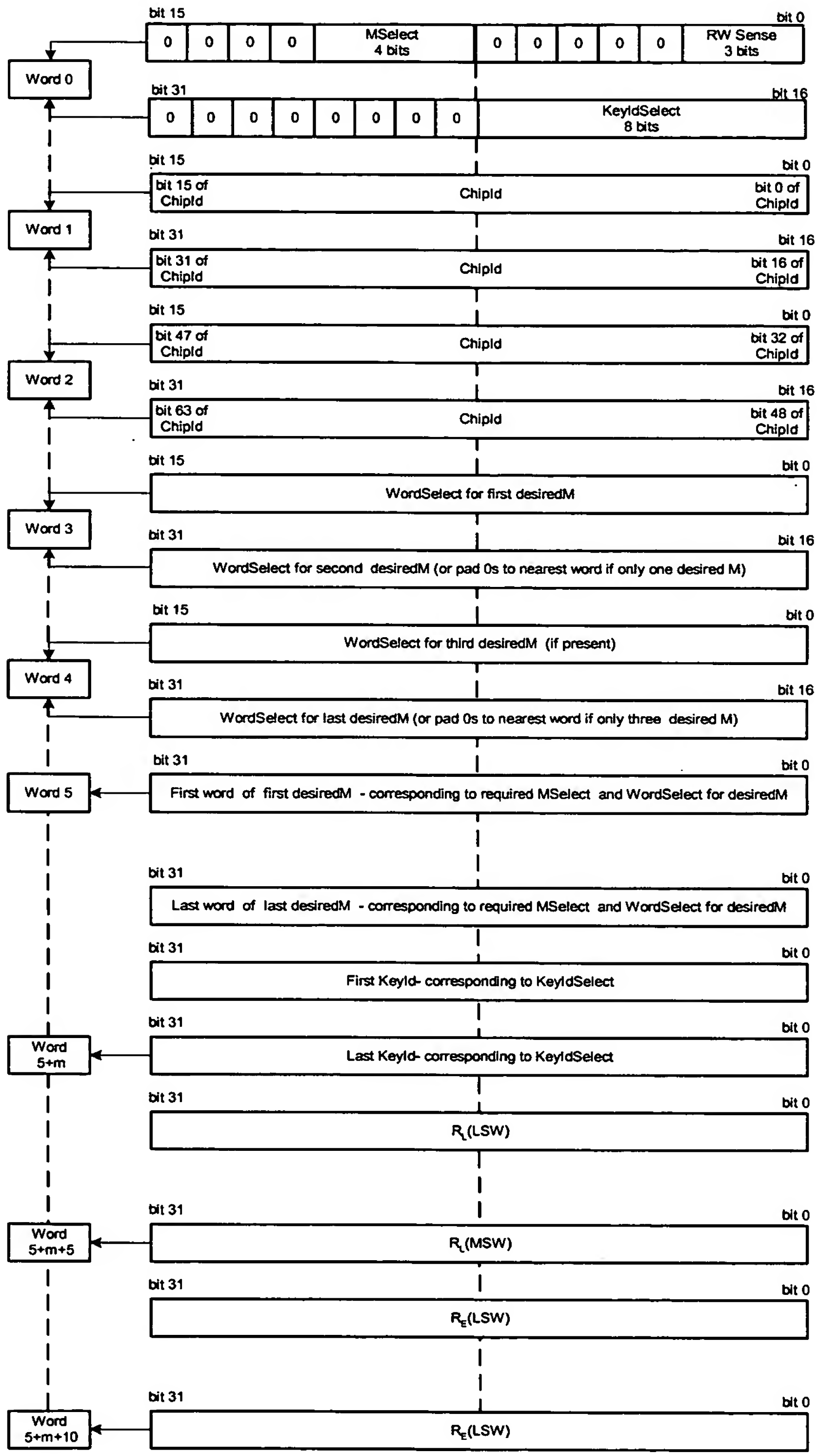

Refill Device QACo
Heirarchical Transfer
Heirarchical Transfer
Heirarchical Transfer
Refill Device ComCo#1
Refill Device ComCo#2
Refill Device ComCo#N
Heirarchical Transfer
Heirarchical Transfer
Refill Device OEM#1
Refill Device OEM#2
Refill Device OEM#N
Peer to Peer Transfer
Peer to Peer Transfer
Refill Device OEM#1 #1
Refill Device OEM#1 #2
Refill Device OEM#1 #N
Peer to Peer Transfer
Peer to Peer Transfer
Ink Device #1
Ink Device #2
Ink Device #N

FIG. 380



bit 15
bit 0
0 0 0 0
MSelect 4 bits
0 0 0 0 0
RW Sense 3 bits
Word 0
bit 31
bit 16
0 0 0 0 0 0 0 0
KeyIdSelect 8 bits
bit 15
bit 0
bit 15 of ChipId
ChipId
bit 0 of ChipId
Word 1
bit 31
bit 16
bit 31 of ChipId
ChipId
bit 16 of ChipId
bit 15
bit 0
bit 47 of ChipId
ChipId
bit 32 of ChipId
Word 2
bit 31
bit 16
bit 63 of ChipId
ChipId
bit 48 of ChipId
bit 15
bit 0
WordSelect for first desiredM
Word 3
bit 31
bit 16
WordSelect for second desiredM (or pad 0s to nearest word if only one desired M)
bit 15
bit 0
WordSelect for third desiredM (if present)
Word 4
bit 31
bit 16
WordSelect for last desiredM (or pad 0s to nearest word if only three desired M)
bit 31
bit 0
Word 5
First word of first desiredM - corresponding to required MSelect and WordSelect for desiredM
bit 31
bit 0
Last word of last desiredM - corresponding to required MSelect and WordSelect for desiredM
bit 31
bit 0
First KeyId- corresponding to KeyIdSelect
bit 31
bit 0
Word 5+m
Last KeyId- corresponding to KeyIdSelect
bit 31
bit 0
R_L(LSW)
bit 31
bit 0
Word 5+m+5
R_L(MSW)
bit 31
bit 0
R_E(LSW)
bit 31
bit 0
Word 5+m+10
R_E(LSW)

FIG. 381

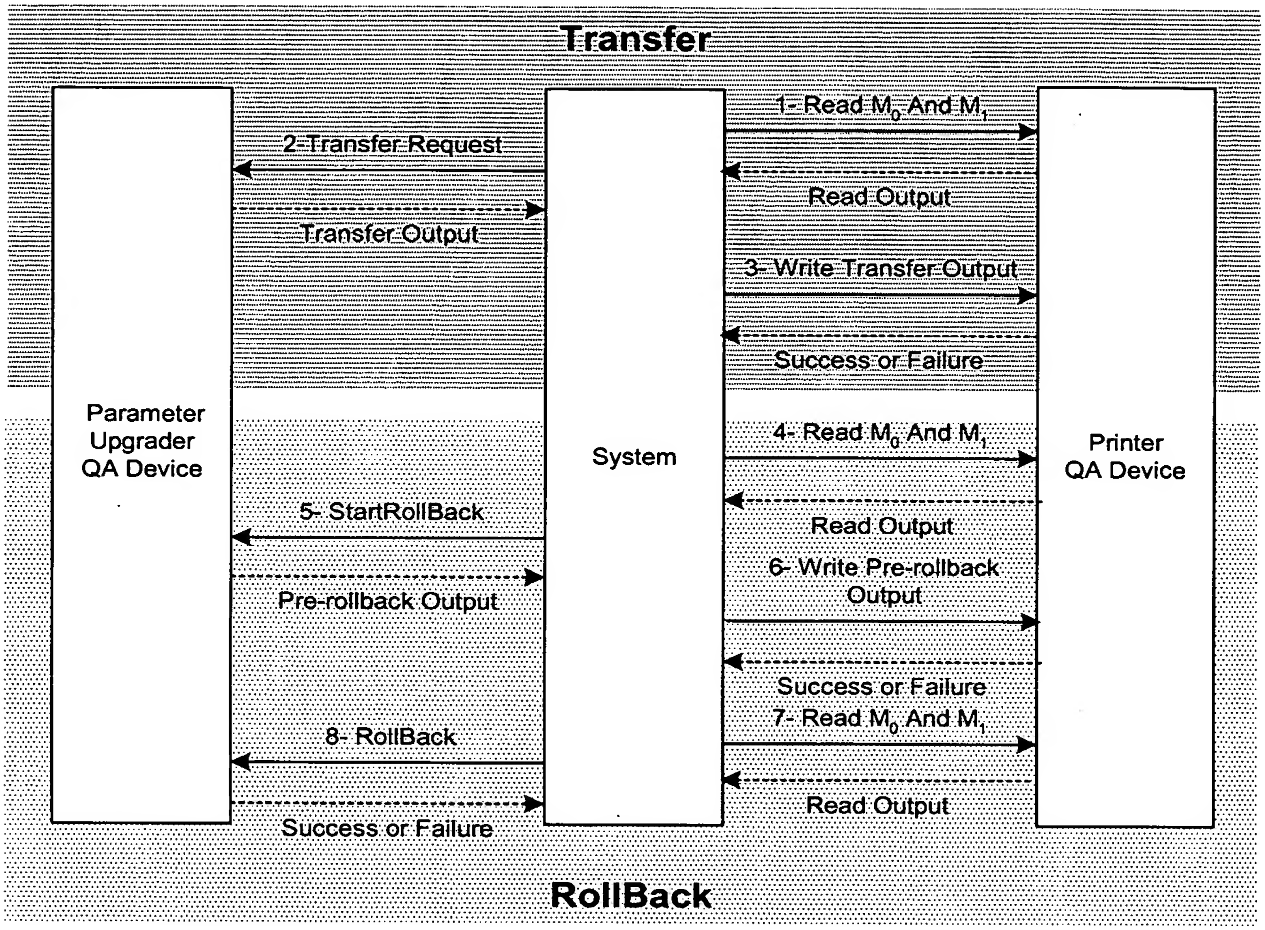
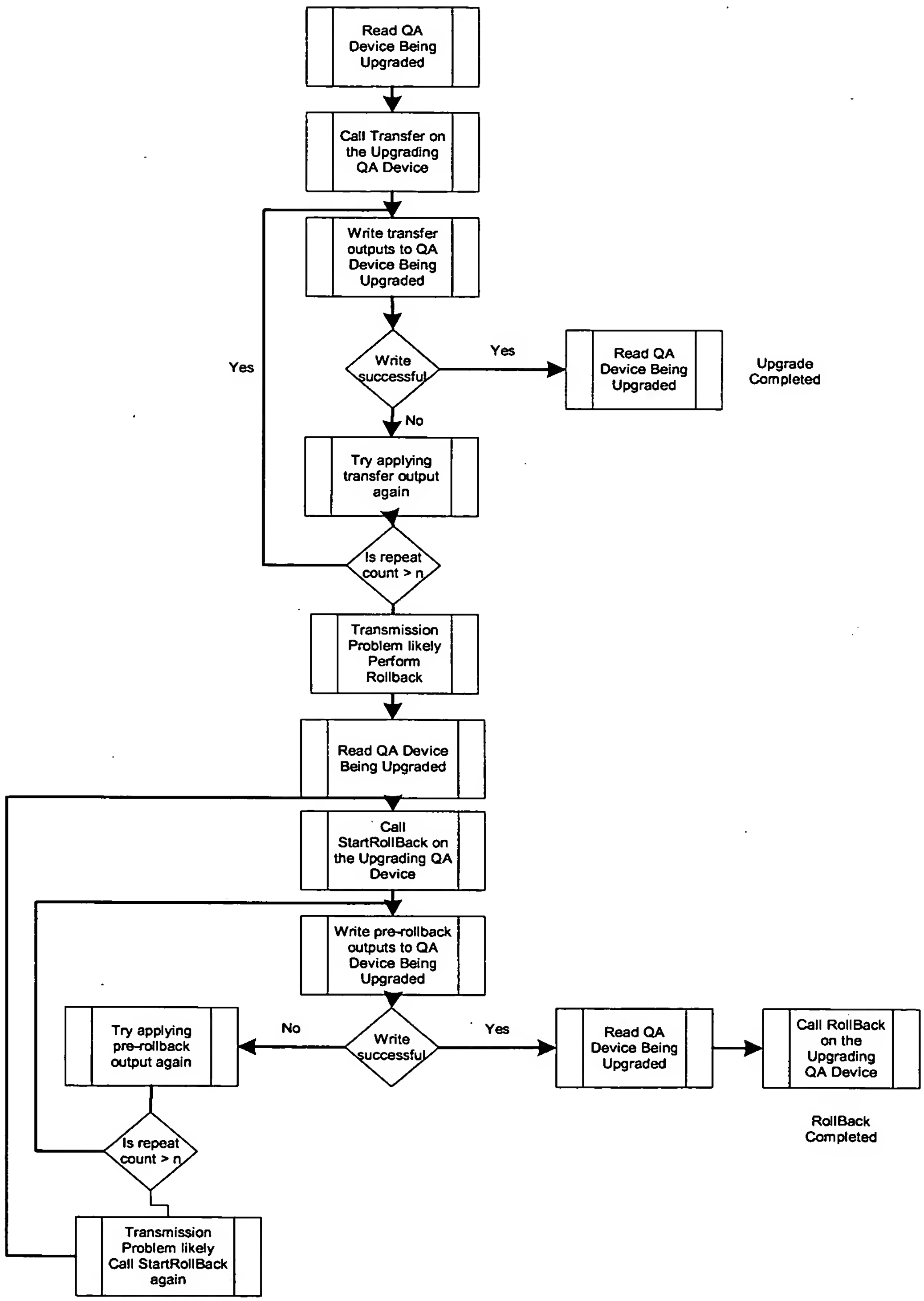

Transfer
Parameter Upgrader QA Device
System
Printer QA Device
1- Read M_0 And M_1
Read Output
2-Transfer Request
Transfer Output
3- Write Transfer Output
Success or Failure
4- Read M_0 And M_1
Read Output
5- StartRollBack
Pre-rollback Output
6- Write Pre-rollback Output
Success or Failure
7- Read M_0 And M_1
Read Output
8- RollBack
Success or Failure
RollBack

FIG. 382



Read QA Device Being Upgraded
Call Transfer on the Upgrading QA Device
Write transfer outputs to QA Device Being Upgraded
Write successful
Yes
Read QA Device Being Upgraded
Upgrade Completed
No
Try applying transfer output again
Yes
Is repeat count > n
Transmission Problem likely Perform Rollback
Read QA Device Being Upgraded
Call StartRollBack on the Upgrading QA Device
Write pre-rollback outputs to QA Device Being Upgraded
Write successful
No
Yes
Try applying pre-rollback output again
Read QA Device Being Upgraded
Call RollBack on the Upgrading QA Device
RollBack Completed
Is repeat count > n
Transmission Problem likely Call StartRollBack again

FIG. 383

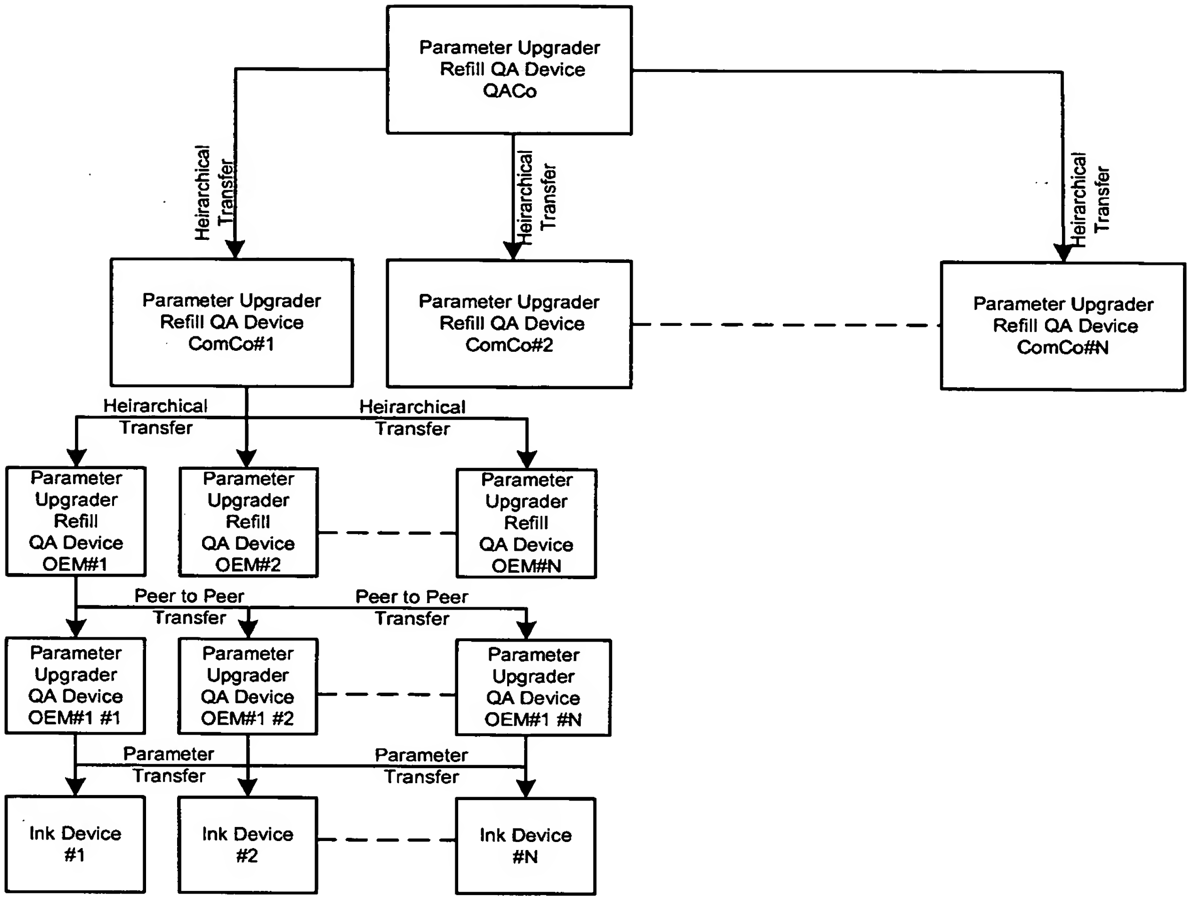


FIG. 384

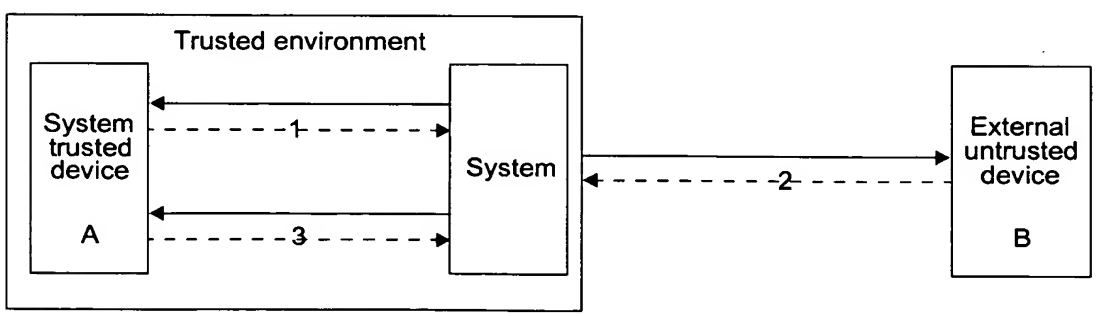


FIG. 385

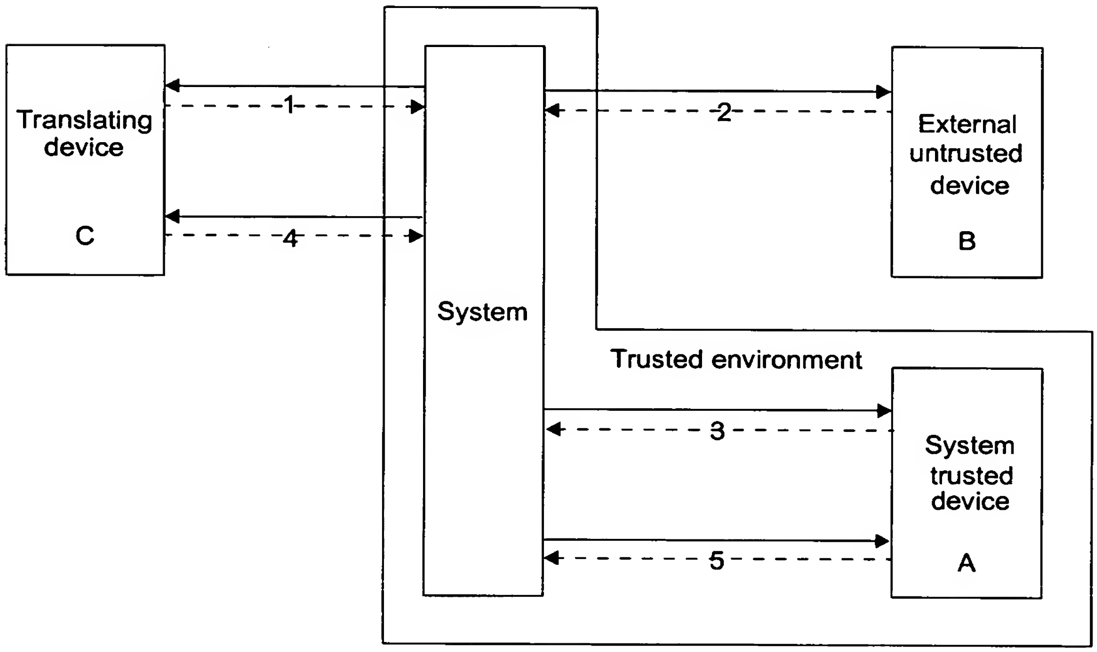

Translating device
C
System
External untrusted device
B
Trusted environment
System trusted device
A
1
2
3
4
5

FIG. 386

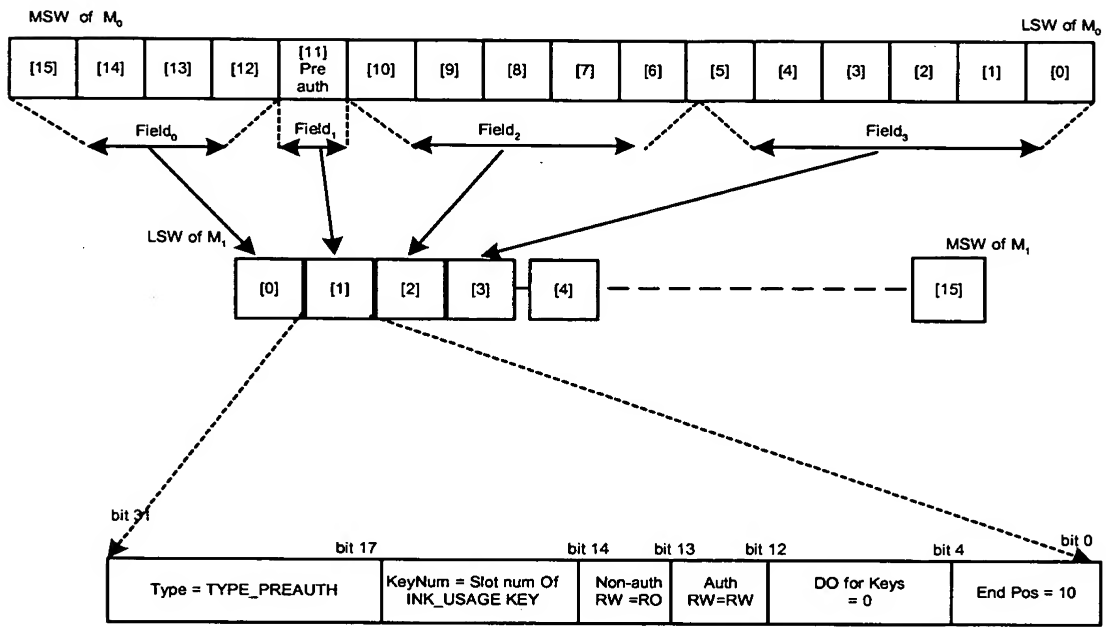


FIG. 387

bit 31	bit 0
preauth field select = 8 bits	preauth amount = 24 bits

FIG. 388

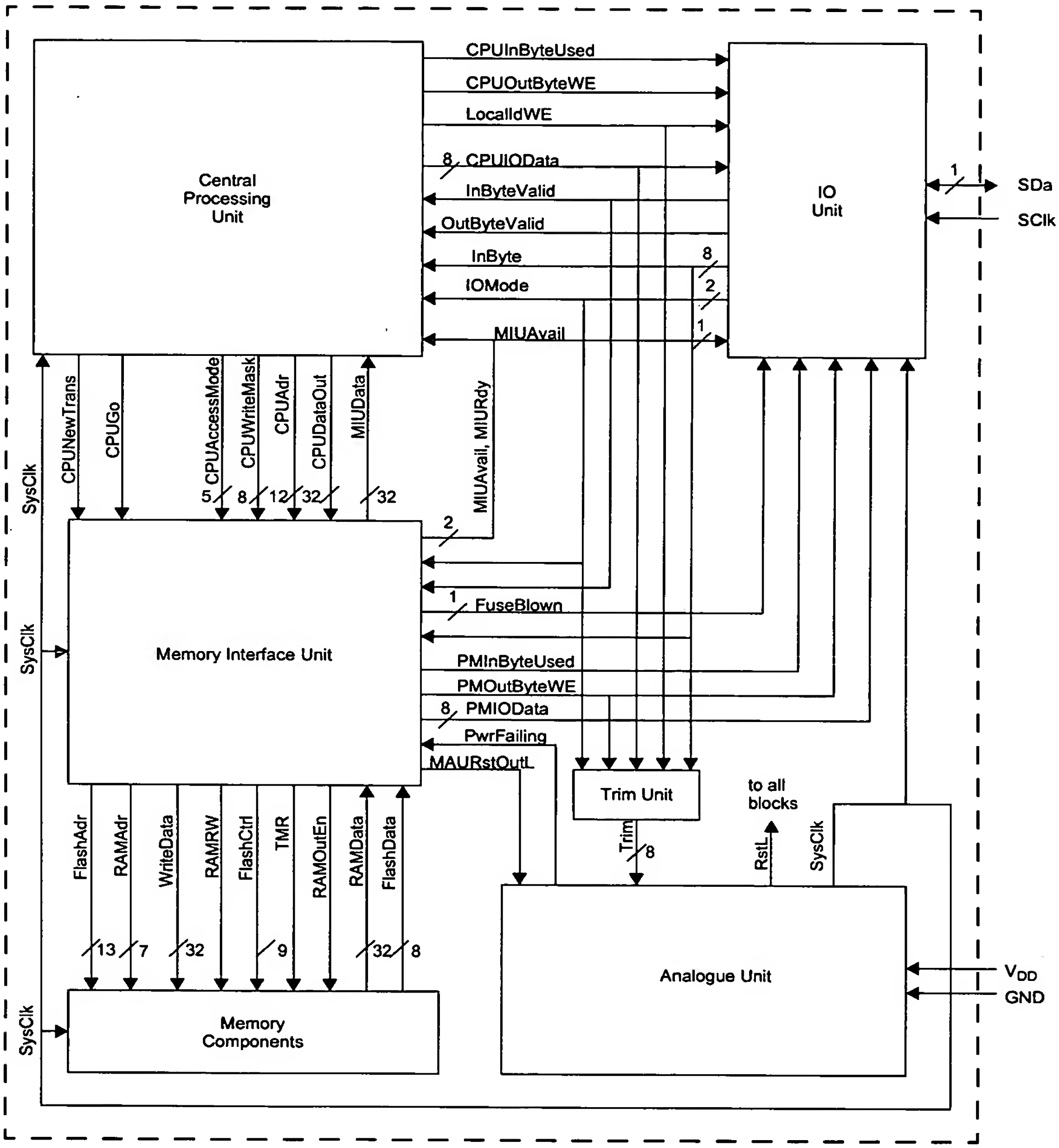

Central
Processing
Unit
IO
Unit
CPUInByteUsed
CPUOutByteWE
LocalIdWE
8 CPUIOData
InByteValid
OutByteValid
InByte
IOMode
MIUAvail
1
SDa
SClk
8
2
1
SysClk
CPUNewTrans
CPUGo
CPUAccessMode
CPUWriteMask
CPUAdr
CPUDataOut
MIUData
MIUAvail, MIURdy
5
8
12
32
32
2
Memory Interface Unit
1 FuseBlown
PMInByteUsed
PMOutByteWE
8 PMIOData
PwrFailing
MAURstOutL
Trim Unit
to all
blocks
Trim
8
RstL
SysClk
FlashAdr
RAMAdr
WriteData
RAMRW
FlashCtrl
TMR
RAMOutEn
RAMData
FlashData
13
7
32
9
32
8
Memory
Components
Analogue Unit
V_{DD}
GND

FIG. 389

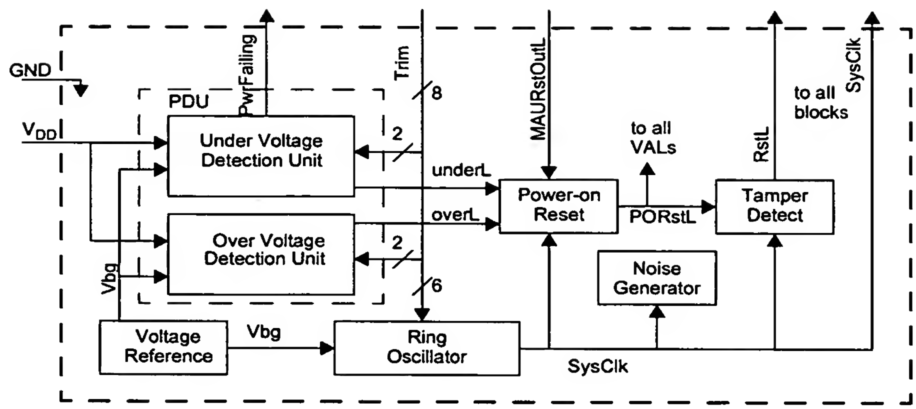


FIG. 390

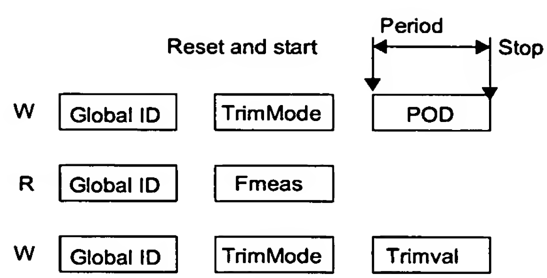


FIG. 391

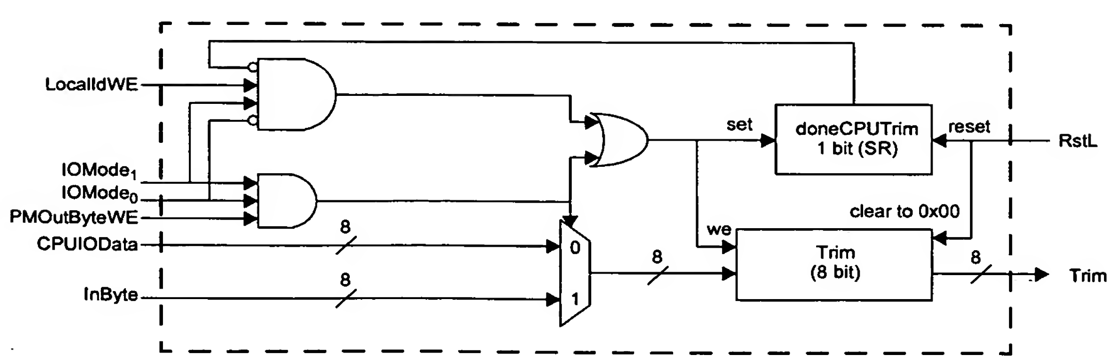


FIG. 392

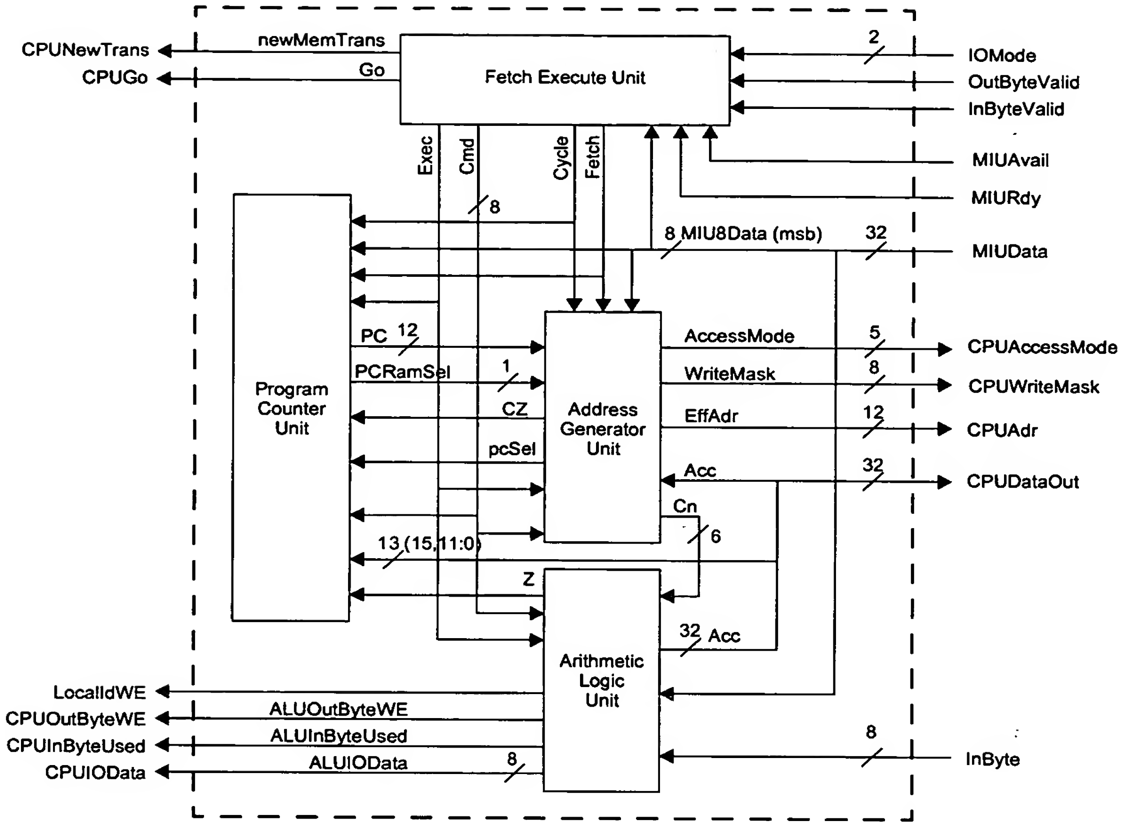

CPUNewTrans
CPUGo
newMemTrans
Go
Fetch Execute Unit
2
IOMode
OutByteValid
InByteValid
MIUAvail
MIURdy
Exec
Cmd
Cycle
Fetch
8
8 MIU8Data (msb)
32
MIUData
Program Counter Unit
PC 12
PCRamSel
1
CZ
pcSel
Address Generator Unit
AccessMode
5
CPUAccessMode
WriteMask
8
CPUWriteMask
EffAdr
12
CPUAdr
Acc
32
CPUDataOut
Cn
6
13 (15,11:0)
Z
Arithmetic Logic Unit
32 Acc
LocalIdWE
CPUOutByteWE
ALUOutByteWE
CPUInByteUsed
ALUInByteUsed
CPUIOData
ALUIOData
8
8
InByte

FIG. 393

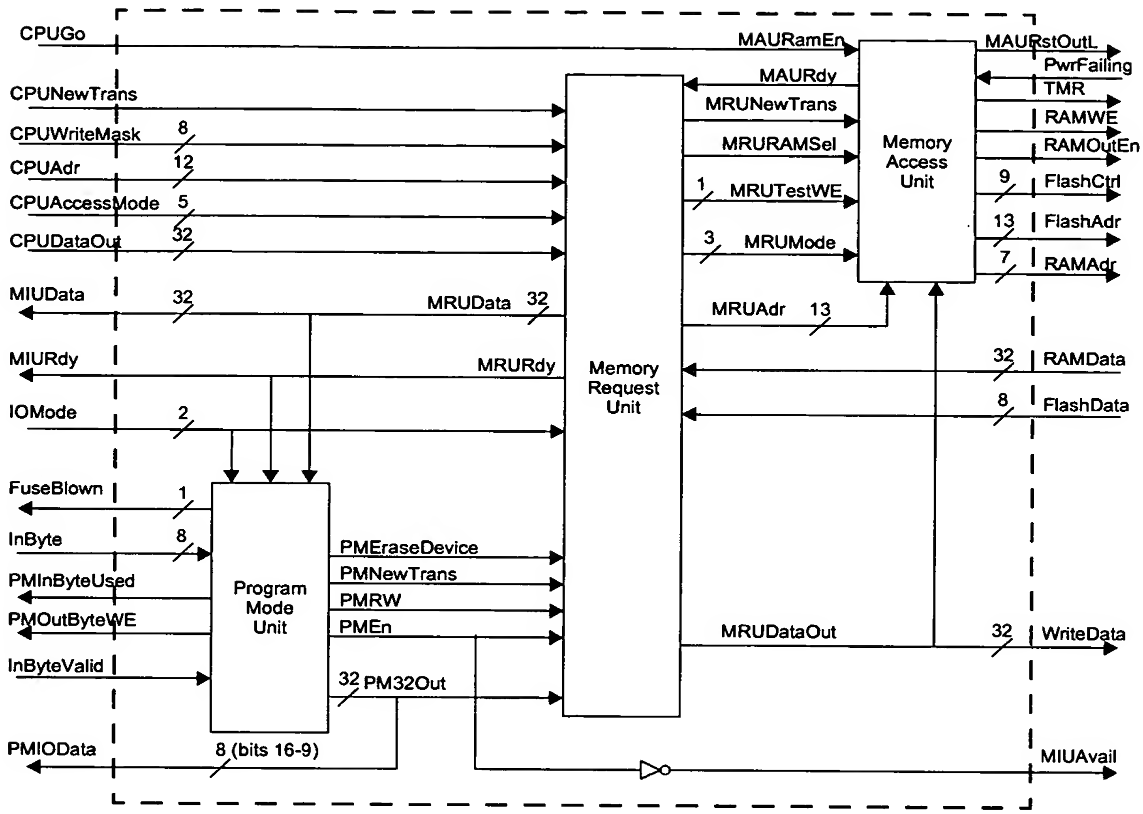


FIG. 394

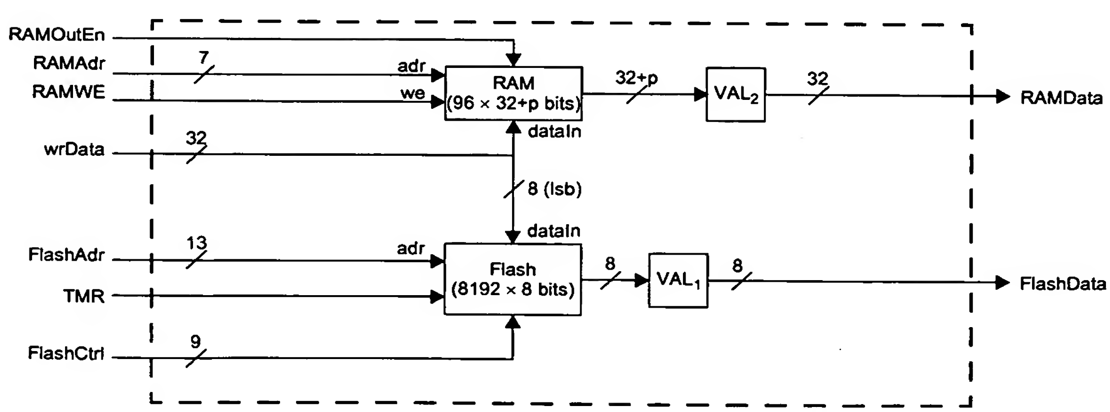


FIG. 395

Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
PriID6	PriID5	PriID4	PriID3	PriID2	PriID1	PriID0	R/*W 0 = write 1 = read

FIG. 396

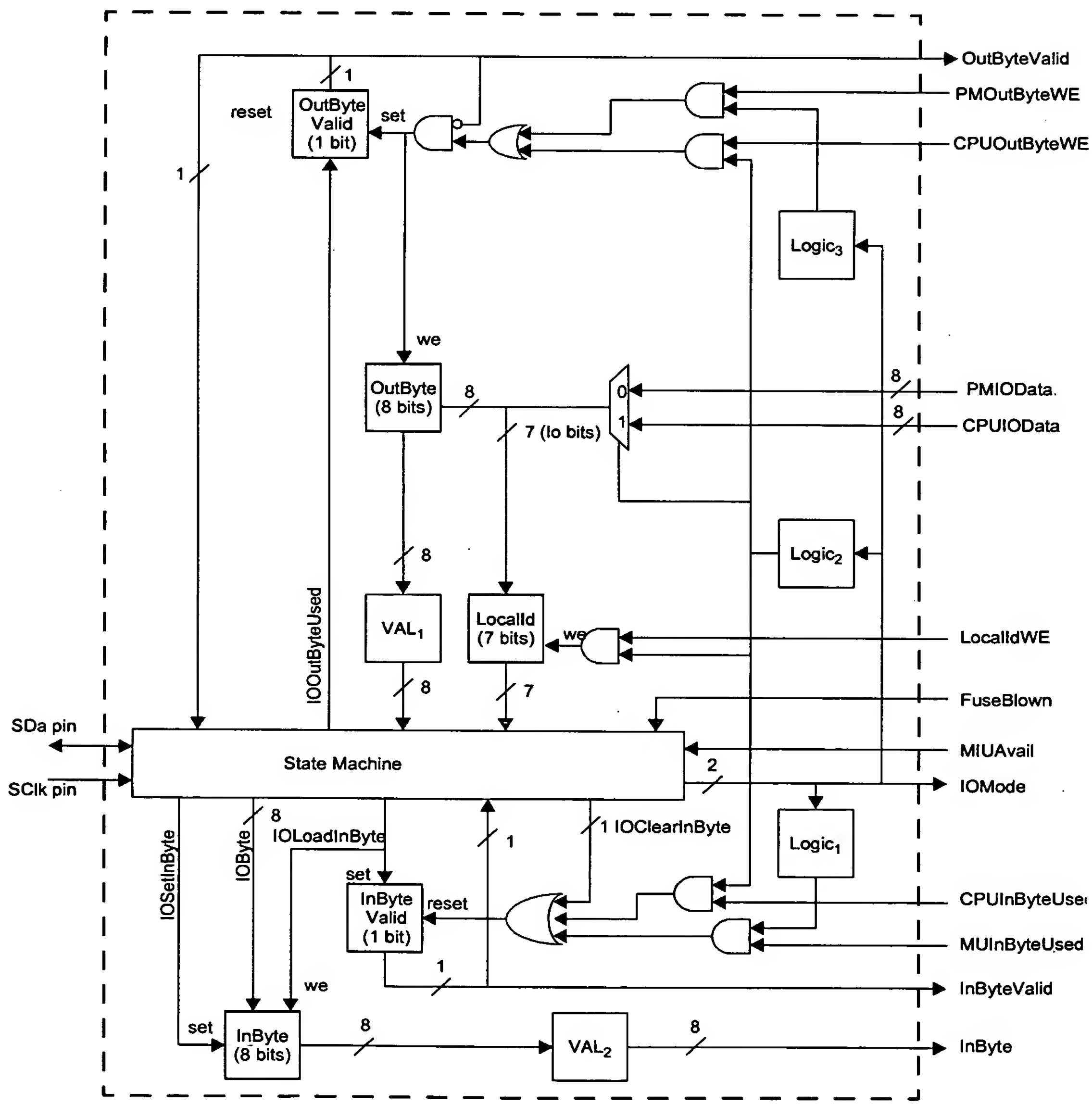


FIG. 397

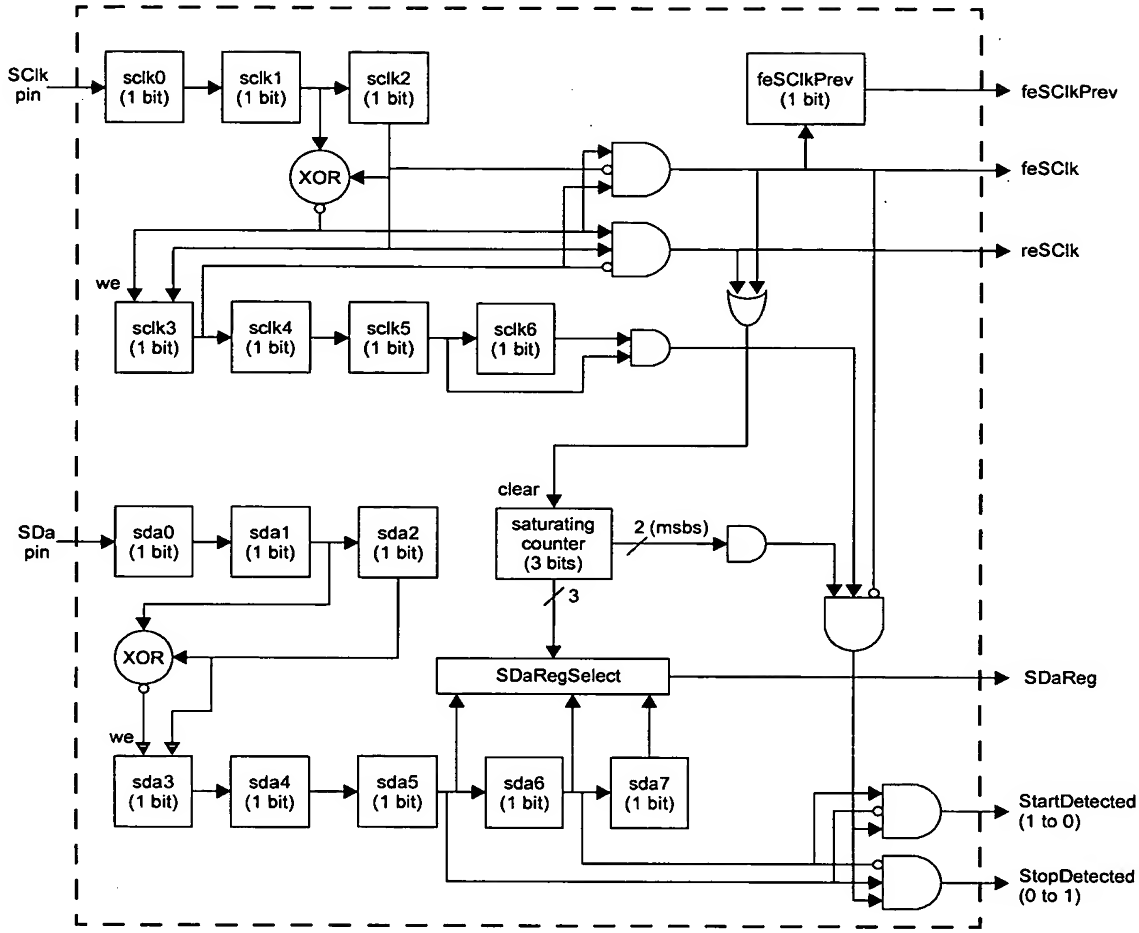

SClk pin
sclk0 (1 bit)
sclk1 (1 bit)
sclk2 (1 bit)
XOR
we
sclk3 (1 bit)
sclk4 (1 bit)
sclk5 (1 bit)
sclk6 (1 bit)
feSClkPrev (1 bit)
feSClkPrev
feSClk
reSClk
clear
saturating counter (3 bits)
2 (msbs)
3
SDa pin
sda0 (1 bit)
sda1 (1 bit)
sda2 (1 bit)
XOR
we
sda3 (1 bit)
sda4 (1 bit)
sda5 (1 bit)
sda6 (1 bit)
sda7 (1 bit)
SDaRegSelect
SDaReg
StartDetected (1 to 0)
StopDetected (0 to 1)

FIG. 398

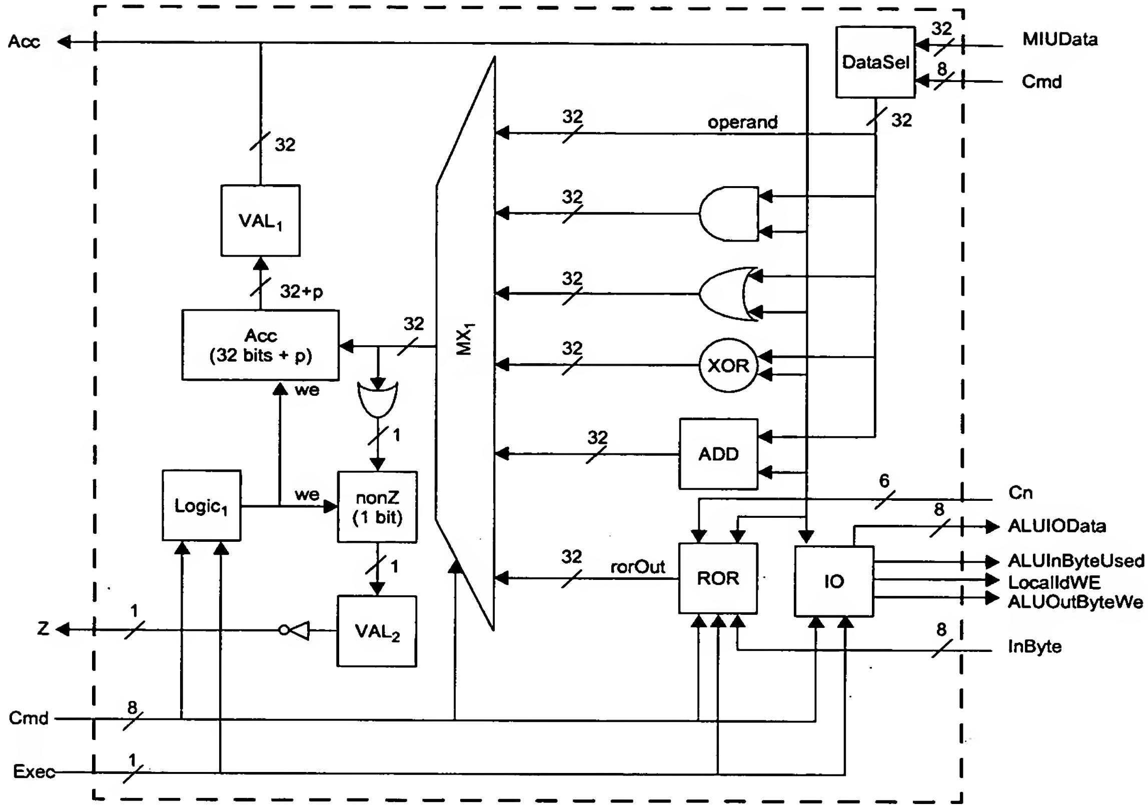


FIG. 399

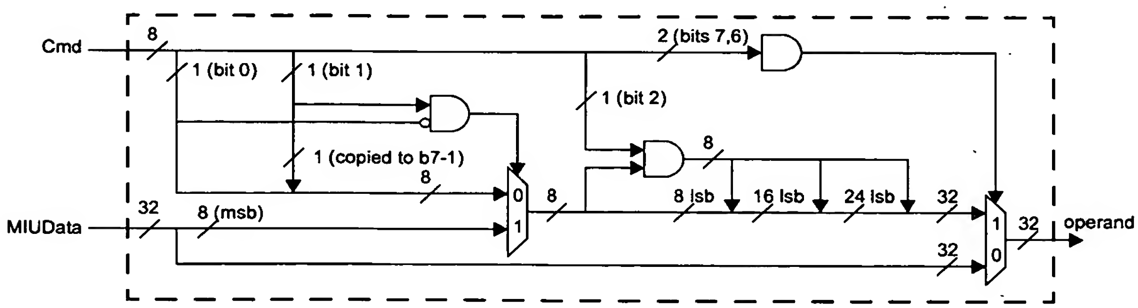
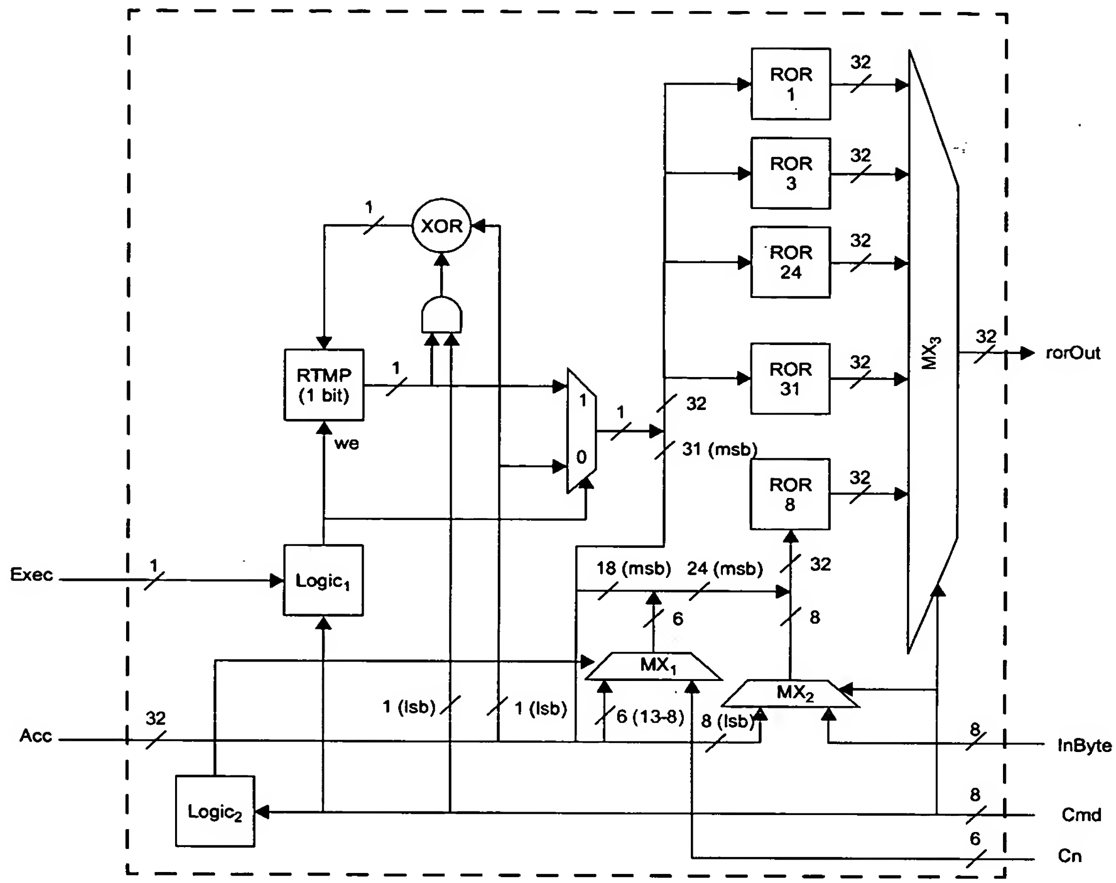
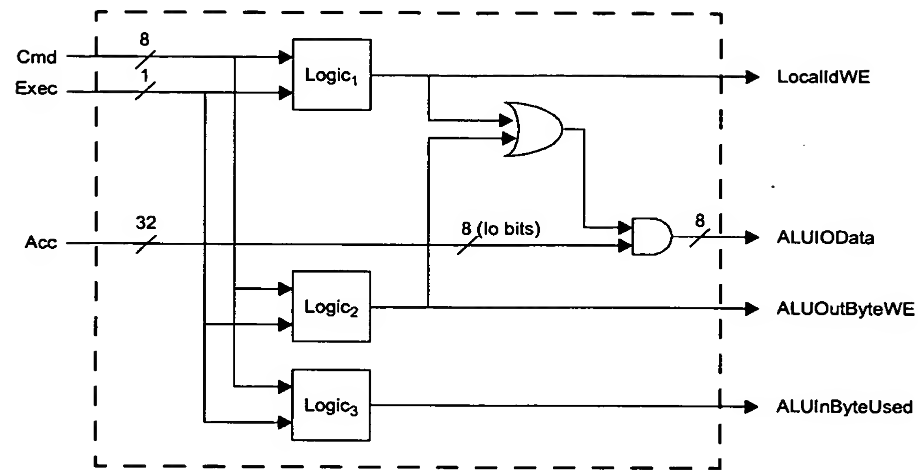


FIG. 400



ROR 1
ROR 3
ROR 24
ROR 31
ROR 8
32
XOR
RTMP (1 bit)
we
1
0
31 (msb)
18 (msb)
24 (msb)
6
8
MX1
MX2
MX3
rorOut
Exec
Logic1
Acc
1 (lsb)
6 (13-8)
8 (lsb)
InByte
Logic2
Cmd
Cn

FIG. 401



Cmd
Exec
Acc
8
1
32
8 (lo bits)
8
Logic1
Logic2
Logic3
LocalIdWE
ALUIOData
ALUOutByteWE
ALUInByteUsed

FIG. 402

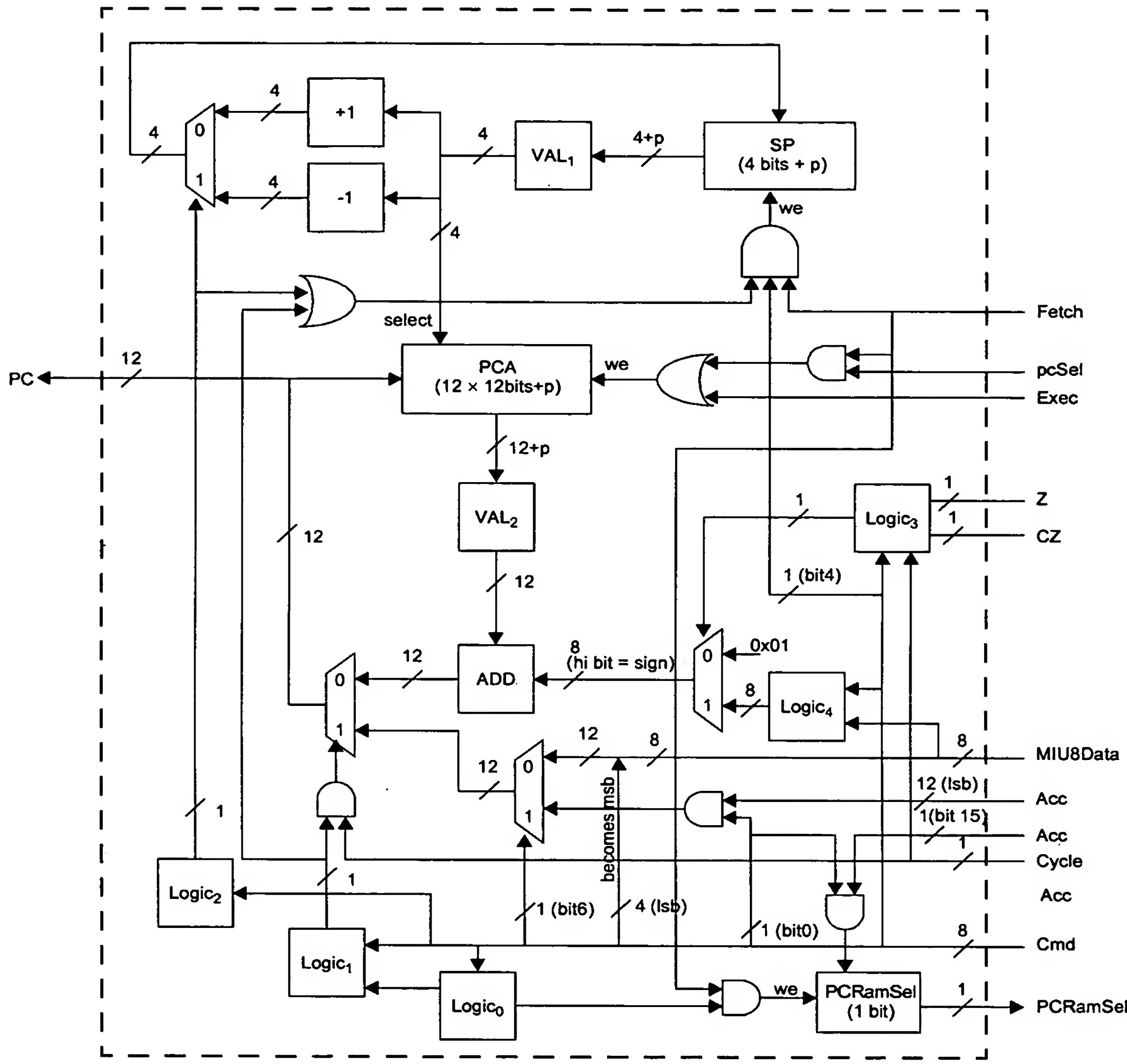

SP
(4 bits + p)
VAL1
+1
-1
PCA
(12 x 12bits+p)
VAL2
ADD
Logic3
Logic4
Logic2
Logic1
Logic0
PCRamSel
(1 bit)
select
we
PC
Fetch
pcSel
Exec
Z
CZ
MIU8Data
Acc
Cycle
Cmd
12+p
0x01
(hi bit = sign)
1 (bit4)
12 (lsb)
1(bit 15)
becomes msb
1 (bit6)
4 (lsb)
1 (bit0)

FIG. 403

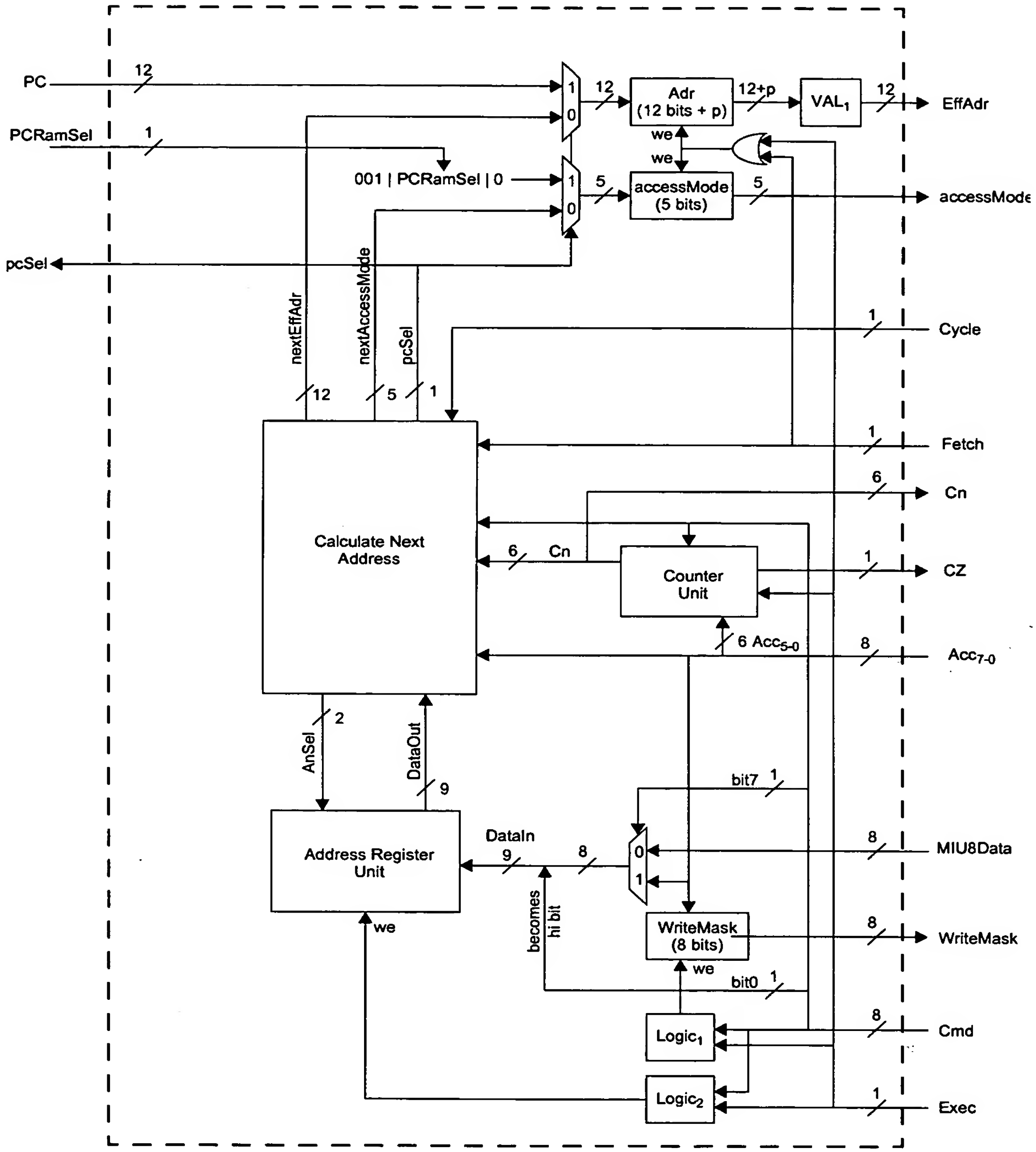

PC
12
PCRamSel
1
001 | PCRamSel | 0
pcSel
Adr
(12 bits + p)
12+p
VAL1
EffAdr
we
we
accessMode
(5 bits)
5
accessMode
nextEffAdr
nextAccessMode
pcSel
Cycle
Fetch
Cn
CZ
Acc7-0
Calculate Next
Address
Counter
Unit
6 Acc5-0
AnSel
DataOut
2
9
bit7
Address Register
Unit
DataIn
MIU8Data
WriteMask
(8 bits)
WriteMask
becomes
hi bit
bit0
Logic1
Logic2
Cmd
Exec

FIG. 404

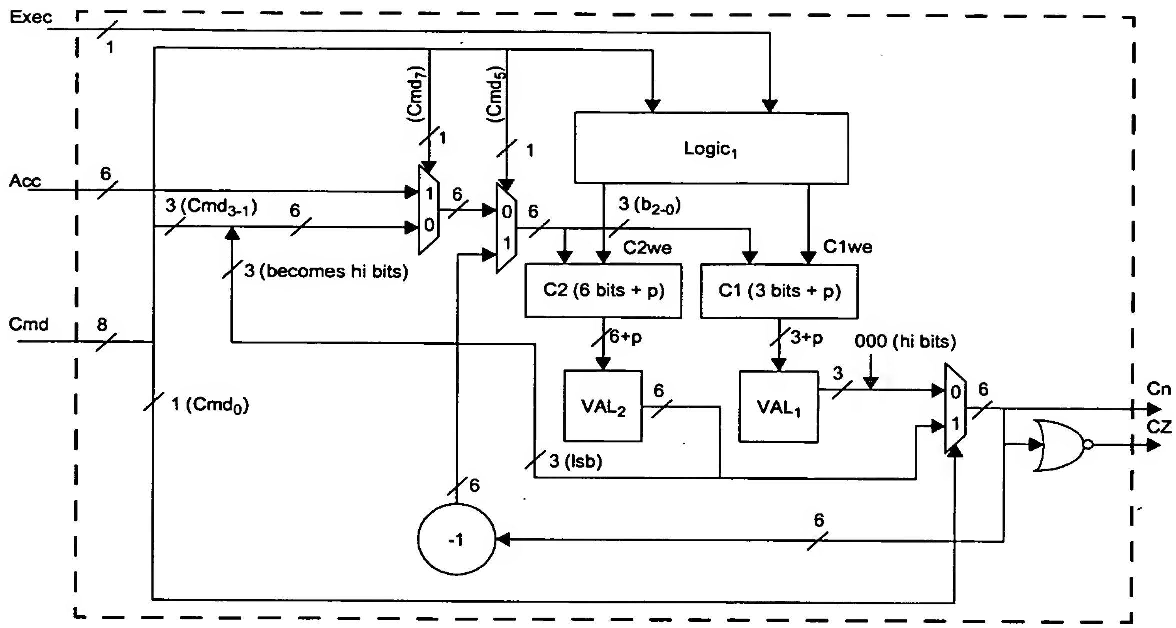


FIG. 405

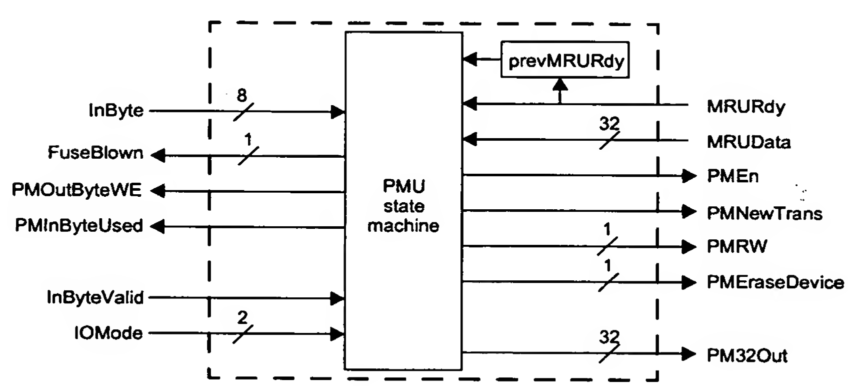


FIG. 406

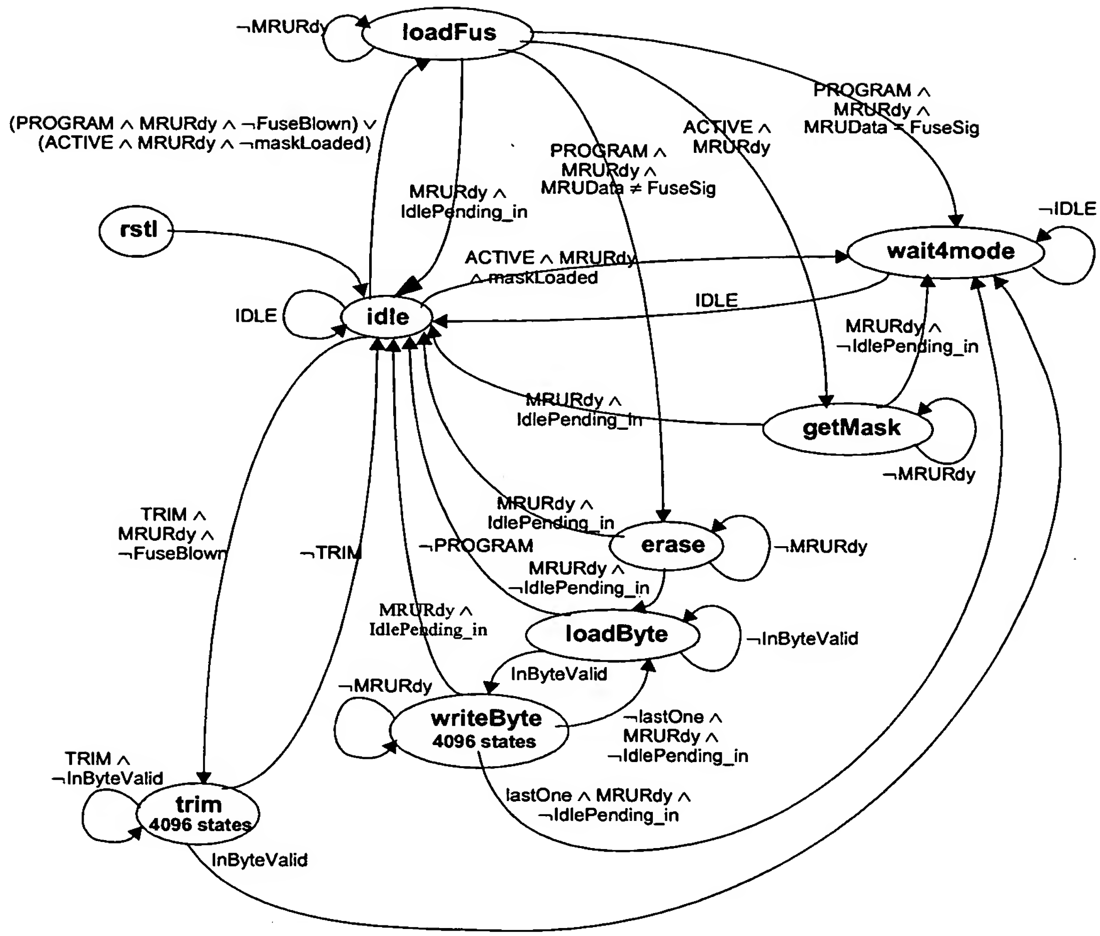

loadFus
¬MRURdy
(PROGRAM ∧ MRURdy ∧ ¬FuseBlown) ∨
(ACTIVE ∧ MRURdy ∧ ¬maskLoaded)
PROGRAM ∧
MRURdy ∧
MRUData = FuseSig
ACTIVE ∧
MRURdy
PROGRAM ∧
MRURdy ∧
MRUData ≠ FuseSig
MRURdy ∧
IdlePending_in
rstl
wait4mode
¬IDLE
ACTIVE ∧ MRURdy
∧ maskLoaded
IDLE
idle
IDLE
MRURdy ∧
¬IdlePending_in
MRURdy ∧
IdlePending_in
getMask
¬MRURdy
TRIM ∧
MRURdy ∧
¬FuseBlown
¬TRIM
¬PROGRAM
MRURdy ∧
IdlePending_in
erase
¬MRURdy
MRURdy ∧
¬IdlePending_in
MRURdy ∧
IdlePending_in
loadByte
¬InByteValid
InByteValid
¬MRURdy
writeByte
4096 states
¬lastOne ∧
MRURdy ∧
¬IdlePending_in
lastOne ∧ MRURdy ∧
¬IdlePending_in
TRIM ∧
¬InByteValid
trim
4096 states
InByteValid

FIG. 407

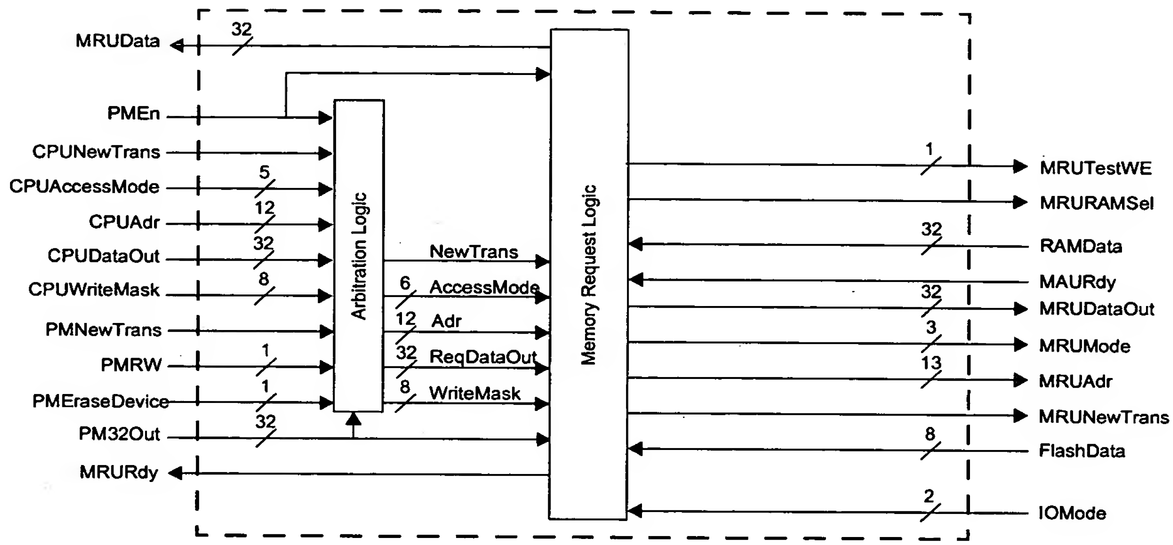

MRUData
32
PMEn
CPUNewTrans
CPUAccessMode
5
CPUAdr
12
CPUDataOut
32
CPUWriteMask
8
PMNewTrans
PMRW
1
PMEraseDevice
1
PM32Out
32
MRURdy
Arbitration Logic
NewTrans
6
AccessMode
12
Adr
32
ReqDataOut
8
WriteMask
Memory Request Logic
1
MRUTestWE
MRURAMSel
32
RAMData
MAURdy
32
MRUDataOut
3
MRUMode
13
MRUAdr
MRUNewTrans
8
FlashData
2
IOMode

FIG. 408

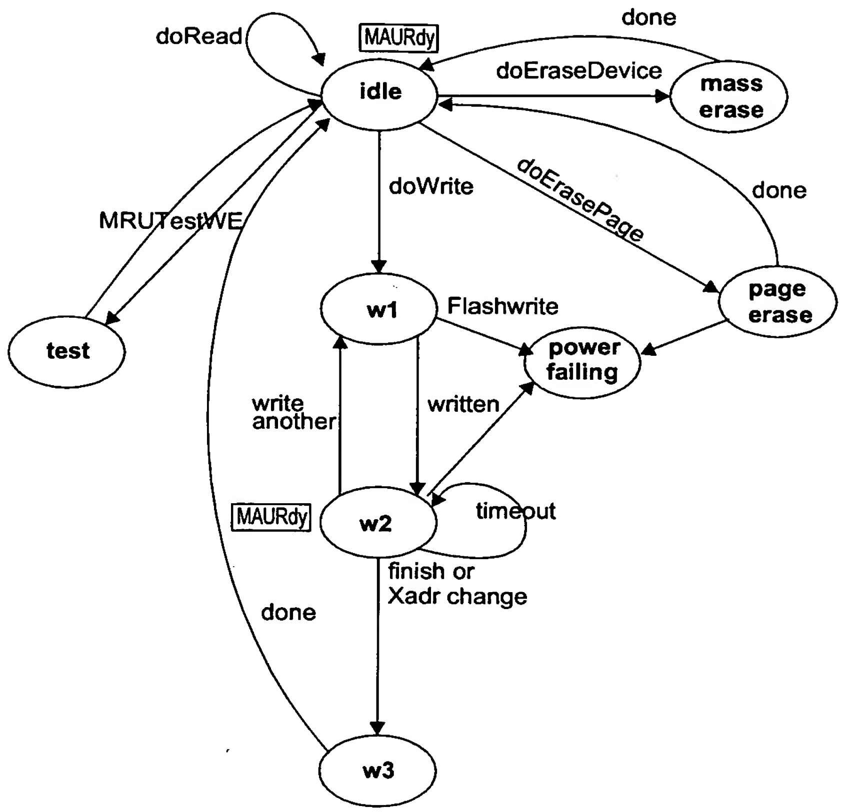


FIG. 409

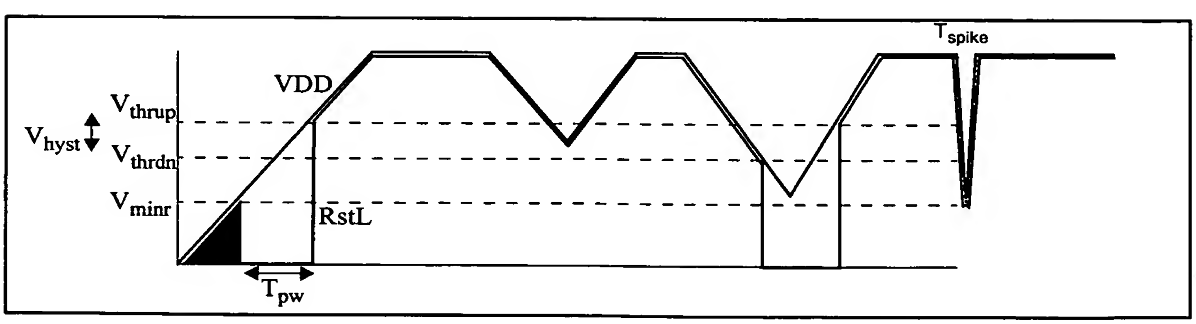


FIG. 410

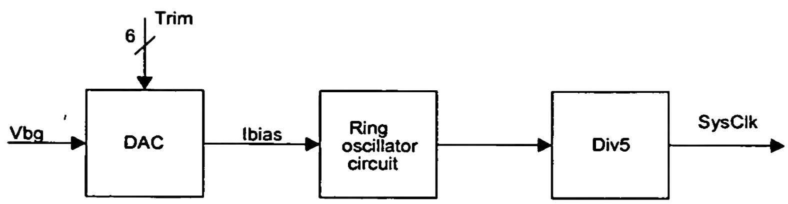


FIG. 411

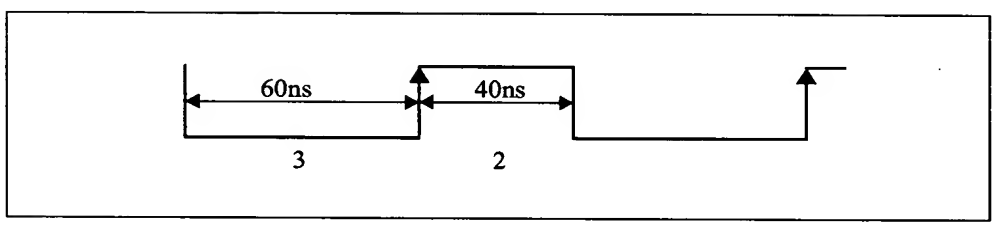


FIG. 412

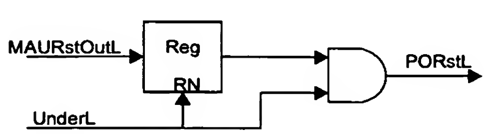


FIG. 413